POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Energia

Corso di Laurea Magistrale in Ingegneria Energetica

Multidimensional simulation of moving geometries with topological changes by an open source CFD code

Relatore: Prof. Federico Piscaglia
Co-relatore: Ing. Andrea Montorfano

Tesi di Laurea Magistrale di:
Filippo Giussani Matr. 799746

Anno Accademico 2014-2015

"Non mi piace il lavoro — non piace a nessuno — ma mi piace ciò che c'è dentro il lavoro — la possibilità di ritrovare te stesso. La tua propria realtà — non per gli altri, ma per te stesso — ciò che nessun altro potrà mai conoscere."
Joseph Conrad, Heart of Darkness

Ringraziamenti

Desidero ringraziare il Prof. Federico Piscaglia e l'Ing. Andrea Montorfano, per la smisurata disponibilità, pazienza e simpatia che hanno dimostrato nei miei confronti ogni volta che sono entrato nei loro uffici, anche in tarda giornata. Ma sopratutto li ringrazio per la conoscenza e la passione che mi hanno trasmesso per la CFD, permettendomi di trovare finalmente la mia strada.

Ringrazio i miei genitori e mio fratello che mi hanno sempre sostenuto durante questi anni, sia nei momenti felici, sia in quelli più difficili, che purtoppo non sono mancati, ma che hanno saputo fortificarmi.

Ringrazio tutte le persone che hanno reso il Poli un posto più piacevole, con cui ho condiviso momenti indimenticabili e di cui avrò tanta nostalgia. Ringrazio tutti i miei compagni di studio per i momenti di gioia e dolore condivisi in questi 5 anni durante le sessioni d'esame infinite.

In fine un ringraziamento a tutti i ragazzi dell'auletta tesisti che hanno reso più piacevole il lavoro di tesi.

Contents

Figures

Tables

Sommario

La necessità di simulare flussi comprimibili turbolenti in geometrie complesse, come motori a combustione interna e ugelli di iniettori, richiede algoritmi capaci di gestire griglie dinamiche con cambiamenti topologici. L'obiettivo di questo lavoro di tesi è sviluppare un'efficiente strategia per la generazione di una griglia di calcolo non strutturata a blocchi, basata su superfici non-conformi e compatibile con OpenFOAM, mediante il codice commerciale ANSYS ICEM CFD. Due casi sono stati analizzati: un motore con pistone a testa piatta con valvola centrata sull'asse di moto del pistone ed un motore a valvole verticali con camera di combustione trasparente. Per entrambi i casi sono state condotte simulazioni mediante il codice di calcolo open-source OpenFOAM, usando un modello di turbolenza ibrido RANS/LES chiamato DLRM. Per il primo caso di studio sono stati simulati otto cicli motore ed i risultati, in termini velocità media assiale e deviazione standard su piani mediani interni al cilindro, sono stati confrontati con i dati sperimentali. Per la seconda geometria è stata simulata una compressione seguita da un'espansione; anche in questo caso, i valori istantanei, calcolati, di pressione e volume del cilindro sono stati confrontati con i dati sperimentali.

Keywords: CFD, OpenFOAM, ICEM, DLRM, coldTopoEngineFoam, topological changes, dynamic mesh

Abstract

The recent need for the simulation of turbulent compressible flows in complex geometries, such as internal combustion engine and injector nozzles, requires algorithms able to handle dynamic grids with topological changes. The aim of this work is the development of an efficient meshing generation strategy by ANSYS ICEM CFD that is suitable for a dynamic mesh handling based on topological changes in OpenFOAM. Two cases have been considered: a flat-top cylinder head engine with a fixed axis-centered valve and a transparent combustion chamber (TCC) engine. Afterward, time-resolved fluid dynamic simulations in OpenFOAM, using the DLRM turbulence model, have been performed. For the first case, mean axial velocity and rms velocity has been compared with LES and experimental measurements. In the second case, only cylinder compression and expansion have been simulated: instantaneous cylinder pressure and volume, calculated, have been compared with experimental ones.

Keywords: CFD, OpenFOAM, ICEM, DLRM, coldTopoEngineFoam, topological changes, dynamic mesh

Riassunto esteso

I dettagli relativi alle condizioni di moto turbolento del fluido nel cilindro di un motore a combustione interna hanno un ruolo fondamentale nella determinazione delle prestazioni del motore e delle relative emissioni. La turbolenza della carica controlla il miscelamento del combustibile con l'ossidante, l'avvio e lo sviluppo del processo di combustione, lo scambio termico. La simulazione numerica dei flussi turbolenti rappresenta uno strumento molto utile per gli ingegneri durante la fase di progetto del motore in supporto alle metodologie sperimentali, poichè quest'ultime comportano elevati costi e maggiori tempi connessi alla realizzazione e taratura di apparati per le campagne di misura.

Per questo motivi, la fluidodinamica computazionale è uno strumento sempre più applicato nella progettazione dei motori degli anni recenti. Storicamente, i flussi turbolenti sono stati simulati principalmente attraverso metodi basati sulla media di Reynolds delle equazioni di Navier-Stokes (RANS), sia nella loro formulazione instazionaria (URANS). Nelle simulazioni condotte sui motori a combustione interna, l'approccio URANS ha dimostrato di poter garantire delle buone predizioni del flusso mediato, fornendo con una buona accuratezza le caratteristiche macroscopiche del moto della carica nel cilindro come i vortici di "swirl" e di "tumble". Tuttavia, la maggior parte dele grandezze fisiche tempo-varianti che caratterizzano i flussi in un cilindro non possono essere risolte da un modello basato su metodi che forniscono valori mediati.

La turbolenza su piccolo scale, la dispersione ciclica e l'evoluzione delle strutture tridimensionali all'interno dei motori possono essere simulate solo con approcci come "Large eddy simulation" (LES). Tuttavia l'alto costo computazionale richiesto dalle simulazioni LES, il complesso settaggio dei casi, la complessa analisi dei dati unita alla necessità di griglie di calcolo molto raffinate, ha spinto verso l'implementazione di modelli ibridi, che in letteratura prendono il nome di "Very Large Eddy Simulation" (VLES). Alla famiglia di questi modelli appartiene il modello di turbolenza usato in questo lavoro: il "Dynamic Length Resolution Model" (DLRM) può disinguere in relazione alla risoluzione spaziale della griglia ed all'avanzamento temporale, quali siano le scale che possono essere risolte e quali debbano essere modellate, producendo risultati con un costo computazionale ragionevole se confrontato a qullo della LES.

Anche il modo con cui la griglia di calcolo è ralizzata e il tipo tesso di griglia contribuiscono sia a ridurre i costi computazionali, sia al perseguimento di risultati af-

fidabili. Infatti, dal momento che i motori possiedono geometrie abbastanza complesse e variabili nel tempo, la griglia deve essere interfacciata con solutori appositamente sviluppati per gestire il movimento del dominio. In questo lavoro, il modello ibrido RANS/LES DLRM è stato usato in combinazione con un nuovo solutore fluidodinamico e ad una nuova strategia per la gestione della griglia dinamica, che ha richiesto una originale strategia di generazione della griglia stessa. In particolare, in questo lavoro, è stata condotta un'analisi approfondita su come realizzare la griglia per garantirne la qualità durante tutta la simulazione dinamica.

Il software fluidodinamico utilzzato in questo lavoro è OpenFOAM (Open Field Operation And Manipulation); OpenFOAM è un codice open-source basato su librerie C++ per la simulazione dei fenomeni fisici collegati alla meccanica del continuo. Il software include diversi solutori per la simulazione di problemi di fluidodinamica classica, utilities e librerie pre-configurate.

Obiettivo di questo lavoro è stato lo sviluppo di una nuova strategia per realizzare griglie di calcolo ad alta qualità per motori a combustione interna, attraverso il software commerciale ANSYS ICEM CFD, che possa soddisfare alcuni requisiti fondamentali:

- avere un appropriato boundary layer vicino alle pareti;
- avere un numero di celle più basso di quello tradizionalmente impiegato in griglie per una simulazione LES;
- una struttura a blocchi che ne faciliti la gestione dinamica nel solver;

Il solutore fluidodinamico utilizzato in questo lavoro e' `coldTopoEngineFoam`, estensione del già esistente solutore instazionario per flussi comprimibili su griglie dinamiche presente nel codice OpenFOAM, con alcune modifiche per il miglioramento della convergenza con regioni multiple che dinamicamente si connettono e disconnettono e flussi in condizioni di blocco sonico attraverso le valvole. Insieme al modello di turbolenza (DLRM) e alle applicazioni utilizzate in questo lavoro, `coldTopoEngineFoam` è incluso all'interno delle librerie di OpenFOAM sviluppate dal gruppo Motori del Politecnico di Milano.

Nei primi capitoli di questo lavoro di tesi verranno prima introdotti i due casi di motori studiati:

1. un motore a valvola fissa centrata rispetto all'asse e pistone in movimento;
2. un motore a testa piatta con valvole verticali e pistone in movimento;

Per i due casi, sono mostrati i criteri e le strategie adottate per la generazione della griglia:

1. gestione e modifica della geometria;

2. gestione e modifica dei blocchi attorno alla valvola ed alla candela, con approccio top-down, che si possa applicare anche su valvole inclinate e altre geometrie di candela;

3. generazione di una griglia non-strutturata a blocchi completamente esaedriche, con applicazione al caso di un motore con valvole verticali.

4. esportazione e manipolazione della griglia generata in formato compatibile con OpenFOAM.

Nell'ultimo capitolo, si mostrano i risultati delle simulazioni fluidodinamiche "time-resolved" condotte in OpenFOAM mediante l'applicazione del solver `coldTopoEngineFoam`, della strategia di mesh dinamica e del modello di turbolenza DLRM sviluppati al Politecnico di Mialno. Per il primo motore, la velocità media e l'intensità di turbolenza nelle direzioni assiale e radiale sono state confrontate con i risultati di simulazioni LES condotte precedentemente e con i risultati sperimentali. Per la seconda geometria, sono state simulate solo la fase di compressione e di espansione del cilindro, mostrando confronti con i dati sperimentali in termini di pressione e di volume istantaneo.

Chapter 1

Introduction

Among all physical phenomena that occur inside an engine cylinder, turbulence has certainly a direct impact on thermodynamic efficiency, brake power and emissions of the engine since its influence extends from volumetric efficiency to air/fuel mixing, combustion and heat transfer. Historically, turbulent flows have been simulated mainly by models based on Reynolds Averaging of Navier-Stokes equations (RANS), either in their original version or in the unsteady formulation (URANS) for slowly-varying flows. In ICE simulation, URANS approaches have proved to provide very good predictions of phase-averaged flow fields: macroscopic features of charge motion like swirl and tumble vortexes can be estimated with a good accuracy. On the other hand, most of the time-varying quantities characterizing in-cylinder flows cannot be resolved by a model based on implicit time or ensemble-averaging methods like URANS: small-scale turbulence, cycle-to-cycle variability (CCV) and in-cycle evolution of three-dimensional structures (jets and vortexes) can be simulated only by a time-resolved (rather than time-averaged) approach like Large Eddy Simulation (LES). However the high computational cost, the complex case-set-up and pre- and post-processing together with the need of unaffordable grid resolution near walls to appreciate LES scales, leads to developed adequate Hybrid model (RANS/LES) which is able to automatically understand when switching between LES and URANS. There is obviously the need in IC engines simulation for a turbulence model that can distinguish, automatically run-time, what can be resolved and what cannot: an hybrid model could be the best way to achieve this purpose and can also provides an easy set-up of case and produces results in a reasonable computational time. An additional problem during simulation of these complex geometries is handle in the best possible way the topological changes due to the moving boundaries. Since ICE are not a static system, to simulate their real behaviour, the motion of boundaries must be taken into account. That need leads to development of algorithms must be able to handle moving domain, new mesh strategy of these geometries and, finally, algorithms able to adjust the behaviour bring about by moving mesh. OpenFOAM (Open Field Operation And Manipulation) is the computational fluid dynamic software used in this simulation. OpenFOAM is a toolbox based on C++

libraries, that is able to simulate numerically all the physical phenomena related to the mechanics of continuous. In particular, it allows the following applications: reactive fluid dynamics of complex fluid, turbulent flow with heat transfer, conjugated calculations and solid mechanics. The software is provided with several pre-configured solvers, utilities and libraries and it can be used as any other typical commercial package for numerical simulation. However, unlike the commercial codes, OpenFOAM is open, not only in terms of source code, but even in its hierarchical structure and design, in such a way that all its solvers, utilities and libraries can be modified according to the needs of the user. The turbulence model (DLRM), solvers (coldTopoENgineFoam) and dynamic mesh handling tools used in this work has been developed within the libICE, an extended version of OpenFOAM library developed by ICE group of Politecnico di Milano for OpenFOAM-2.3.x. The purpose of my work has been the development of a new meshing strategy, through the commercial software ANSYS ICEM CFD, that perfectly suits the geometry of ICE and its sub-parts like valves and spark plug as well and that is easily repeatable on similar geometries. In particular the mesh has to fulfil some requirements:

- appropriate layering near wall;
- a number of cell lower than of a mesh used for LES simulation.
- a block structure in ICEM which allows for a dynamic mesh handling in OpenFOAM.

Hence the work is organized according to the following pattern. Firstly the physics of the problem and the numerical approach to model are presented. Then an in-depth analysis of how the mesh is generated in ICEM is presented, from geometry/blocking creation and adjustment up to mesh generation and smoothing techniques.
In particular two cases have been studied: Flat-top cylinder head with a fixed axis-centred valve and Transparent Combustion Chamber case (TCC). Interfacing ICEM mesh with OpenFOAM has been dealt as well. Afterwards simulation set-up of two cases aforementioned is discussed, in order to explain boundary condition, numerical schemes, numerical solution and algorithm control. Subsequently the numerical results are shown in several images that describe the comparison between numerical (LES/DLRM) and experimental results. Finally, in the conclusions section an evaluation of the work and future developments are presented.

Chapter 2

Governing equation and the finite volume method for moving domain

2.1 Governing equations

The Fluid behaviour could be described by a system of three equations: continuity equation, momentum equation and energy equation. An equation of state is added to them and, due to the fact that air is the working fluid, the equation of state is considered as ideal gas. The Navier-Stokes equations for fluid are written following the Eulerian approach, therefore the variables (p,u,T, etc.) and fluid proprieties (ρ, μ, etc.) are expressed as a function of space and time, and their balance is evaluated on a fixed volume of space traversed by the fluid. In this case the fluid motion is described by a system of partial differential equations. Another way to write the N-S equations is the Lagrangian approach, where the volume on which to write the balance is deformable and in motion with the fluid itself. The Eulerian Approach is made possible by the theorem of transformation (or Leibniz), which allows to write correctly, even for a fixed space control volume, the substantial derivative that is formulated for a volume integral with the body in motion. There is not a right or wrong approach, but it depends on the typology of problem to solve. In this case it is preferred the Eulerian approach because it is possible to separate time dependence from spatial dependence while, using the Lagrangian approach, even in presence of steady state phenomena, dependence on time remains because it is connected to the integration volume that must be followed and which varies in function of time.

Thus it is possible to write the conservation equation of the generic physical property φ, defining V as control volume delimited by boundary surface S, n as the surface normal vector of S, $\mathbf{u}$ as the fluid velocity and Q_φ as the generic source of φ.

$$\frac{d}{dt}\int_V \rho\varphi(\mathbf{x},t)dV = \int_V Q_\varphi dV \tag{2.1}$$

Where the first member of Eq.2.1 represents the total variation in time of φ:

$$\frac{d}{dt}\int_V \rho\varphi(\mathbf{x},t)dV = \frac{\partial}{\partial t}\int_V \rho\varphi(\mathbf{x},t)dV + \int_S \rho\varphi(\mathbf{x},t)\cdot\mathbf{u}\cdot\mathbf{n}dS \tag{2.2}$$

Thus equation 2.1 can be written as:

$$\frac{\partial}{\partial t}\int_V \rho\varphi(\mathbf{x},t)dV + \int_S \rho\varphi(\mathbf{x},t)\cdot\mathbf{u}\cdot\mathbf{n}dS = \int_V Q_\varphi dV \tag{2.3}$$

This general formulation does not allow to find the value of φ in all points of the domain. Referring to the Gauss theorem in order to move from surface integrals to volume integrals:

$$\int_S \rho\varphi(\mathbf{x},t)\cdot\mathbf{u}\cdot\mathbf{n}dS = \int_V \nabla\cdot(\rho\varphi\mathbf{u})dV \tag{2.4}$$

Hence Eq. 2.3 becomes:

$$\frac{\partial}{\partial t}\int_V \rho\varphi(\mathbf{x},t)dV + \int_V \nabla\cdot(\rho\varphi\mathbf{u})dV = \int_V Q_\varphi dV \tag{2.5}$$

2.1.1 Continuity equation

Continuity equation can be obtained by putting in equation 2.5, $\varphi = 1$ and $Q_\varphi = 0$ because there are not source terms for the mass, then:

$$\frac{\partial}{\partial t}\int_V \rho(\mathbf{x},t)dV + \int_V \nabla\cdot(\rho\mathbf{u})dV = 0 \tag{2.6}$$

Which represents the mass conservation of a compressible fluid.

2.1.2 Momentum equation

Momentum equation of a compressible fluid can be written as:

$$\frac{\partial}{\partial t}\int_V \rho\mathbf{u}dV + \int_V \nabla\cdot(\rho\mathbf{u}\mathbf{u})dV = \int_S \boldsymbol{\sigma}\mathbf{n}dS + \int_V \rho\mathbf{f}dV \tag{2.7}$$

Which can be rewritten with Gauss theorem as:

$$\frac{\partial}{\partial t}\int_V \rho\mathbf{u}dV + \int_S \rho\mathbf{u}\mathbf{u}\cdot\mathbf{n}dS = \int_V \boldsymbol{\sigma}dV + \int_V \rho\mathbf{f}dV \tag{2.8}$$

where $\mathbf{f}$ represents body forces acting on the fluid contained in the control volume and $\boldsymbol{\sigma}$ is the stress tensor. The stress tensor depends on two contributions, viscous and pressure effects and it can be defined as:

$$\sigma_{ij} = (-p + 2\lambda\nabla\cdot\mathbf{u}\delta_{ij} + \tau_{ij}) \tag{2.9}$$

where δ_{ij} is Kronecker's delta and λ the second viscosity. τ_{ij} is the viscous stress tensor, which depends on the fluid type. Introducing Newtonian fluid hypothesis, stress tensor can be rewritten as:

$$\tau_{ij} = 2\mu S_{ij} + \lambda S_{ii}\delta_{ij} \tag{2.10}$$

where S_{ij} is the rate of strain tensor defined as:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.11}$$

the hydrostatic part of stress tensor can be written as:

$$\frac{1}{3}\sigma_{ii} = -p + \lambda S_{ii} + \frac{2}{3}\mu S_{ii} \tag{2.12}$$

2.1.3 Energy equation

Energy equation can be written in enthalpy form. Defining specific enthalpy as:

$$h = u + \frac{p}{\rho} \tag{2.13}$$

and introducing Fourier's Law for heat transfer by conduction:

$$q = -k\nabla T \tag{2.14}$$

where k is thermal diffusivity. Finally, energy equation can be written as:

$$\begin{aligned}\frac{\partial}{\partial t}\int_V \rho\Big(h + \frac{u^2}{2}\Big)dV + \int_V \rho\nabla\cdot\mathbf{u}\Big(h + \frac{u^2}{2}\Big)dV + \\ - \int_V \frac{\partial p}{\partial t}dV - \int_S \alpha\nabla h\cdot\mathbf{n}dS = \int qdV\end{aligned} \tag{2.15}$$

Where α is the thermal diffusivity.

2.1.4 Equation of state

To close the system of equation, one equation of state is necessary. If the fluid is considered as an ideal gas , the ideal gas law can be used:

$$Pv = R^*T \tag{2.16}$$

where $R^* = R/MM$. Otherwise if the fluid is far away from ideal behavior other more complex equation must be used i.g. Van der Waals' law or Redlich-Kwong-Soave.

2.2 Governing equations for moving geometries

The conservation equations are usually formulated for static boundary but the recent need to describe turbulent flows in complex geometries, especially with moving boundaries, leads to rewrite equations taking into account the motion of the domain. If the control volume is not constant within the time due to a moving boundary, the only change in conservation equation will be the appearance of relative velocity in convective terms. Thus Eqs. 2.6 2.8 2.15 becomes respectively:

$$\frac{\partial}{\partial t}\int_{V(t)} \rho(\mathbf{x},t)dV + \int_{S(t)} \nabla\rho(\mathbf{u}-\mathbf{u}_b)\cdot\mathbf{n}dS = 0 \tag{2.17}$$

$$\frac{\partial}{\partial t}\int_{V(t)} \rho\mathbf{u}dV + \int_{S(t)} \rho\mathbf{u}(\mathbf{u}-\mathbf{u}_b)\cdot\mathbf{n}dS = \int_{V(t)} \boldsymbol{\sigma}dV + \int_{V(t)} \rho\mathbf{f}dV \tag{2.18}$$

$$\begin{aligned}\frac{\partial}{\partial t}\int_{V(t)} \rho\Big(h+\frac{u^2}{2}\Big)dV + \int_{S(t)} \rho\cdot\Big(h+\frac{u^2}{2}\Big)(\mathbf{u}-\mathbf{u}_b)\cdot\mathbf{n}dS + \\ -\int_{V(t)}\frac{\partial p}{\partial t}dV - \int_{S(t)}\alpha\nabla h\cdot\mathbf{n}dS = \int_{V(t)} qdV\end{aligned} \tag{2.19}$$

where $\mathbf{u}_b$ is the velocities the control volume boundaries move with.

The finite volume method

Equations system described in section 2.1 are solved with numerical methods. The solution method used is the Finite Volume Method (FVM) [1] based on the solution of the equations in integral form over the computational domain previously discretized in a finite number of control volumes, which constitute the mesh. Referring to a governing equation for generic property φ:

$$\frac{\partial}{\partial t}\int_V \rho\varphi dV + \int_S \rho\varphi\cdot\mathbf{u}\cdot\mathbf{n}dS = \int_S \Gamma\nabla\varphi\cdot\mathbf{u}\cdot\mathbf{n}dS + \int_V Q_\varphi dV \tag{2.20}$$

where Γ represents φ diffusivity and Q_φ is a generic source term. The first step consists in the discretization of the computational domain in a finite number of control volumes, which constitute the mesh. At the center of control volume the computational node is defined and it is calculated as center of gravity of the volume, on which equations are solved. The variables in these points represent the cell mean value. By adding the equations of all the cells, the equation 2.20 is obtained. Since the contributions of the integrals on the internal faces cancel out each other, just the integral remains on the boundary, while the contributions of volume integrals add up to a global single term. In this way the conservation property is guaranteed on each volume, as well as it is

guaranteed at a global level over the whole domain. A three-dimensional grid is used to discretize the domain. A typical control volume is shown in Figure 2.1:

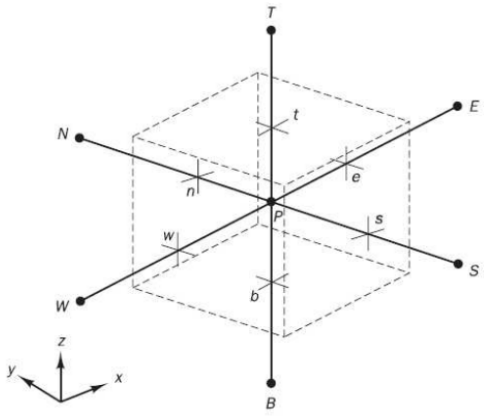


Figure 2.1. A cell in three dimension and neighboring nodes on which discretization is done

Chapter 3

Turbulence modeling and solver

After a brief introduction about the problem encountered in simulating turbulence flows inside ICE a description of novel hybrid RANS/LES turbulence model is dealt. Afterwards a description of fluid dynamic solver used in OpenFOAM is given and problem about its coupling with topology changes is discussed.

3.1 Turbulence

Most flows encountered in engineering practice are turbulent and therefore require different treatment. Turbulent flows are characterized by the following properties:

- they are highly unsteady;
- they are three-dimensional;
- they contain a great deal of vorticity. Indeed, vortex stretching is one of the principal mechanisms by which the intensity of turbulence is increased;
- turbulence increases the rate at which conserved quantities are stirred. Stirring is a process in which parcels of fluid with differing concentrations of at least one of the conserved properties are brought into contact. The actual mixing is accomplished by diffusion. This process is called turbulent diffusion;
- by means of the processes just mentioned, turbulence brings fluids of differing momentum content into contact. The reduction of the velocity gradients due to the action of viscosity reduces the kinetic energy of the flow; in other words, mixing is a dissipative process. The lost energy is irreversibly converted into internal energy of the fluid;
- it has been shown that turbulence flows contain coherent repeatable structures and essentially deterministic events that are responsible for a large part of the

mixing. However, the random component of turbulent flows causes these events that differ form each other in size, strength, and time interval between occurrences, making their study very difficult;

- turbulent flows fluctuate on a broad range of length and time scales. This property makes direct numerical simulation of turbulent flow very difficult.

In particular, for industrial simulations of ICE (internal combustion engine), URANS (unsteady Reynolds average Navier-Stokes) equations are established as a standard tool: the complete turbulence behavior is enclosed within appropriate turbulence model which takes into account all turbulence scales. Turbulence length and time scales are estimated by dimensional considerations, and model transport equations are solved on reasonably coarse grids, that makes this approach relatively cheap in terms of computational cost. Nevertheless RANS is able to give a reasonable approximation of the wall shear stress, macroscopic features of charge motion, like swirl and tumble vortexes, with a good accuracy [2], it is quite well known that the excessive predicted viscous behavior very often damps out the unsteady motion and the flow unsteadiness because it overestimates the modeled turbulent length and time scales. Moreover it is generally accepted that URANS solution is completely determined by initial and Boundary Condition (BC), thus cannot account for randomness or independent events in the flow: It is characterized by simulation results which are perfectly repeatable if BC, with same computer, are used for the unsteady computation. However most of the time-varying quantities characterizing in-cylinder flows cannot be resolved by a model based on implicit time or ensemble-averaging methods like URANS: small-scale turbulence, cycle-to-cycle variability (CCV) and in-cycle evolution of three-dimensional structures (jets and vortexes) can be simulated only by a time-resolved (rather than time-averaged) approach like Large Eddy Simulation (LES) In the last decay, an alternative attractive way has been the application of LES (Large Eddy Simulation) in ICE by several authors [3, 4, 5, 6, 7], whose work has shown good predictions of mean and fluctuating velocity, together with estimation of turbulence-driven CCV. Since in LES, the turbulent length scale is related to the computational grid and to the turbulent time scale from the resolved flow, this approach is potentially more accurate and is able to provide the intrinsic unsteadiness of flow. However the required grid resolution may become very expensive and hence computational cost increase severely. For these reason a compromise is given by hybrid models which are all based on the same idea to represent a link between RANS and LES as they use the best feature of two approach: they resolve the turbulence where possible (LES) and model it elsewhere (RANS). Thus, they try to keep computational efficiency of RANS and the potential of LES to resolve the large turbulent structure even on coarser grid and with high Reynolds numbers. In the recent year, several hybrid LES/RANS methods have been proposed such as Detached Eddy Simulation (DES), Limited Numerical Scales (LNSs), partially averaged Navier-Stokes (PANS), scale adaptive simulations (SAS) and Very

Large Eddy Simulation (VLES). Especially VLES starts to expand as a promising compromise for simulation of industrial flow problems and in ICE as well. Its strength is based on how model acts on turbulence spectrum: its smaller part is resolved and the influence of a larger part of the spectrum has to be expressed with the model. Thus VLES requires the definition of an appropriate filtering technique which can distinguish between resolved an modeled part of the turbulence spectrum, which is a good solution for simulation on ICE. The filtering procedure provides the adaptive characteristics of the VLES models, enabling them to be applied for the whole range of turbulence modeling approaches from the RANS to DNS. Thus for the purpose of this work VLES is suitable since promise a reliable results with lower computational effort than a LES, also on coarser grid.

3.1.1 DLRM

The VLES model used here is the Dynamic Length Scale Resolution Model (DLRM) developed by Piscaglia, A.Montorfano, and Onorati [8] and validated on two static case (swirling flow through a sudden expansion , flow around a poppet valve), and then use in [9] to validate it on moving mesh. The model name is due to its capability to dynamically adapt its behavior according to the grid resolution and to consequently switch from modeling to resolving the turbulent length scale. Its filtering approach is converse to LES one since instead of solving the filtered equations, avoiding the computation of small scales, the modeled length and time scales are filtered in order to suppress their negative influence on the unsteady flow field. Similarly to the work of Gyllenram and Nilsson [10], the functional form of the filter is derived from the relation between filtered and non-filtered time scales. In addition was extended to compressible flows and the functional form of the filter was based on Length Scale Resolution (LSR) parameter making it particularly suitable for ICE.

The filtering technique has been applied to a compressible formulation of k-ω SST model including the optional term for rough walls. This choice allows to retain the robustness and accuracy of the RANS model formulation in the near wall regions and in zones of the free-stream region where the mesh resolution is not sufficiently high for the direct solution of the main turbulent scales. The two equations closure model is coupled to RANS by eddy-viscosity assumption:

$$-\tau_{ij} = 2\mu_t S_{ij} - \frac{2}{3}\rho k \delta_{ij} \tag{3.1}$$

where τ_{ij} is the viscous stress tensor and S_{ij} is the strain rate tensor, defined as:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.2}$$

The Boussinesq assumption introduces the concept of a turbulent eddy viscosity μ_t, which is particularly suitable when the influence of turbulence on the mean flow is

dominated by mixing process. The eddy viscosity of the fluid has the same dimension of dynamic viscosity of the fluid and is assumed to be proportional to a function of the turbulent length and time scales:

$$\mu_t \sim \frac{L_t^2}{T_t} \tag{3.3}$$

Where the turbulent length scale L_t and time scale T_t are unknown local properties of the turbulent flow and must be modeled and can be directly evaluated by the turbulent kinetic energy and dissipation rate:

$$L_t \sim k^{1/2}/\omega \tag{3.4}$$

$$T_t \sim 1/\omega \tag{3.5}$$

If the modeled turbulent kinetic energy k and specific dissipation rate ω are solved, a measure of the turbulent length and time scales can be estimated. A low-pass filtering operation, on frequency, is applied to the turbulence model in order to allow the existence of resolvable turbulent scales in the solution of the flow field. The filter function is derived from a dimensional analysis and the filter is applied to the turbulent length and time scales, rather than only to the turbulent kinetic energy.

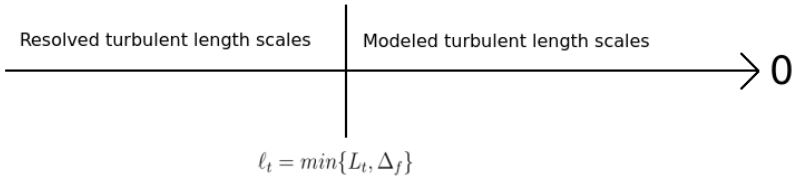


Figure 3.1. Resolved scales and modeled scales

In this model the filtering operation is based on the comparison between the modeled and the resolved turbulent length scales: if the modeled scales are larger than the resolvable scales, resolvable scales will replace the modeled scales in the formulation of the eddy viscosity. As shown in Fig. 3.1 the upper limit of the modeled turbulent length scales corresponds exactly to the lower limit of the resolved turbulent length scales and there is no lower limit because the mean non-resolved turbulent length scale may be much smaller than the local grid spacing, especially close to the walls. Thus the largest length scale that needs to be a part of the eddy viscosity formulation is:

$$\ell_t = min\{L_t, \Delta_f\} \tag{3.6}$$

Where L_t is modeled length scale and Δ_f is resolved length scale defined as follow:

$$\Delta_f = max\{\alpha|\mathbf{U}|\delta t, \Delta_{eq}\} \tag{3.7}$$

whose terms will discuss later.

As aforementioned the filter function is based on the comparison between the modeled and the resolved turbulent length scales. To do that the upper limit of the modeled length ℓ_t and time scales t_t can also be defined in terms of the filtered (non-resolved) variables:

$$\ell_t \sim \hat{k}^{1/2}/\hat{\omega} \tag{3.8}$$

$$t_t \sim 1/\hat{\omega} \tag{3.9}$$

Where the specific dissipation rate modeled ω and filtered $\hat{\omega}$ are:

$$\omega = \frac{\epsilon}{\beta^* k} \tag{3.10}$$

$$\hat{\omega} = \frac{\hat{\epsilon}}{\beta^* \hat{k}} \tag{3.11}$$

Where $\beta^* = 0.09$. As the dissipation rate is never resolved in anything cheaper that a Direct Number Simulation (DNS), hence:

$$\hat{\epsilon} = \epsilon \tag{3.12}$$

Thus this assumption allows to correlate specific dissipation rate modeled and filtered:

$$\hat{\omega} = \frac{\epsilon}{\beta^* \hat{k}} = \frac{\omega k}{\hat{k}} \tag{3.13}$$

Using Eqs. 3.4, 3.8, from the Eq. 3.13 is possible to find an expression for the filtered turbulent kinetic energy:

$$\hat{k} = \frac{\omega k}{\hat{\omega}} = \frac{k^{3/2} L_t}{\hat{k}^{1/2}/\ell_t} \tag{3.14}$$

$$\hat{k} = \left(\frac{\ell_t}{L_t}\right)^{2/3} k = g(\ell_t, L_t)k \tag{3.15}$$

Thus the filter function results to be defined as:

$$g \equiv \left(\frac{\ell_t}{L_t}\right)^{2/3} \tag{3.16}$$

in which ℓ_t is computed from Eq.3.6 and the modeled turbulent scale L_t is calculated as:

$$L_t \simeq \frac{k^{1/2}}{\beta^* \omega} \tag{3.17}$$

A filtered eddy viscosity can be constructed directly from the non-resolvable turbulent length and time scales:

$$\hat{\mu_t} \sim \ell_t^2 / t_t \tag{3.18}$$

It follows from Eqs. 3.8, 3.5, 3.13 and 3.15 that:

$$\hat{\mu_t} = g^2 \rho \frac{k}{\omega} = g^2 \mu_t \tag{3.19}$$

which means that the filter is directly applied to Reynolds stress tensor left unchanged the turbulence model, thus it can be applied to any two-equation model Hence the filter operates in this way:

Condition	ℓ_t	g	$\hat{\mu_t}$	$\hat{k}$
if $L_t < \Delta_f$	L_t	1	μ_t	k
if $L_t > \Delta_f$	$max\{\alpha \lvert \mathbf{U} \rvert \delta t, \Delta_{eq}\}$	$(\frac{\Delta_f}{L_t})^{2/3}$	$g^2 \mu_t$	gk

Table 3.1. Automatic choice of turbulence model based on length scale

From Tab. 3.1 it is possible understand how the filter operates: in regions where turbulence cannot be resolved the filter is equal to unity and the eddy viscosity recovers to the original, non filtered, formulation.

Where turbulence can be resolved the filter is applied and the choice of his value is made between a LES equivalent filter size Δ_{eq} and the product $\alpha |\mathbf{U}| \delta t$ that is a measure of the shortest distance over which a fluid particle can be traced in an unsteady computation and the maximum length scale that needs to be a part of the eddy viscosity formulation. In this sense, the computational time step influences the lower limit for the resolved time scale and includes the implicit relationship between space and time in the filtering operation. Both parameters are calculated dynamically during the simulation:

- $\Delta_{eq} = LSR \cdot \ell_{di}$
  LSR represent a sort of energy distance between the actual resolved energy level and the corresponding lower limit of the inertial sub-range and it is defined as:

$$LSR_{computed} = \frac{\Delta}{\ell_{di}} \tag{3.20}$$

where Δ is the local filter size (usually cubic square of cell volume $(\Delta X \Delta Y \Delta Z)^{1/3}$) and ℓ_{di} is the lower limit of the inertial sub-range:

$$\ell_{di} \approx 60\eta \tag{3.21}$$

and η is the Kolmogorov scale:

$$\eta = \nu^{3/4}\epsilon^{-1/4} \tag{3.22}$$

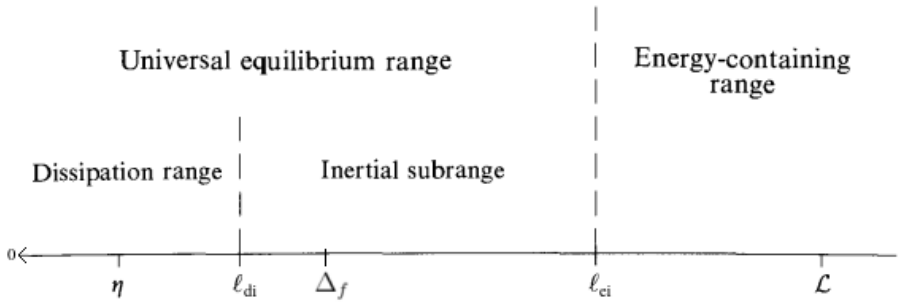


Figure 3.2. Eddy size (on logarithmic scale) at very high Reynolds numbers, showing the various length scales and ranges. The suffixes "di" means that ℓ_{di} is the demarcation line between Dissipation and inertial range while "ei" is the demarcation between Energy and inertial

From the definition of Eq. 3.20, the evaluation of the actual resolved energy level is directly linked to the local filter size. In this way the adopted mesh size is related to the local energy resolution all over the computational domain. In [11], it was found that $LSR \leq 5$ is the upper limit to guarantee a reasonable LES resolution at an affordable computational cost. Thus LSR_{max} is fixed to this value while $LSR_{computed}$ is computed dynamically since mesh size can change and ℓ_{di} change at any time step (using cinematic viscosity of the fluid and assuming that in Eq. 3.22 the dissipation rate is the modeled one).
Thus the LSR used in Δ_{eq} formulation is set to LSR_{max} only if $LSR_{computed}$ assumes a value higher than 5, otherwise is set to $LSR_{computed}$.

- $\alpha|\mathbf{U}|\delta t$ comes from the condition imposed by maximum flow Courant-Friedrichs-Lewy (CFL) number:

$$CFL_{max} = \frac{\mathbf{U}_{max}\delta t}{\delta x_{min}} \tag{3.23}$$

In this way the integration time step is usually limited to:

$$\delta t \leq \frac{CFL_{max}\delta x_{min}}{|\mathbf{U}_{max}|} \tag{3.24}$$

CFL_{max} is usually chosen on the basis of the numerical method and other considerations such as the degree of unsteadiness of the problem. Also each cell at each time step own its local CFL:

$$CFL_i = \frac{\mathbf{U}_i \delta t}{\delta x_i} \tag{3.25}$$

and the criterion 3.24 ensures that for each cell of domain CFL_i is lower than CFL_{max} :

$$\beta = \frac{CFL_i}{CFL_{max}} \leq 1 \tag{3.26}$$

In general, there is no way to mathematically distinguish between turbulence and unsteadiness. For this reason, it is very difficult to estimate the minimum length scale that the turbulence model is able to capture, since it depends both on the spatial resolution and on the temporal correlation between time steps, which is strictly related to the CFL used by the numerical solver. A conservative way of thinking would lead to the conclusion that for each computational cell the smallest turbulent length scale that can be captured by the turbulence model is $|\mathbf{U}\delta t|_{max}$:

$$|\mathbf{U}\delta t|_{max} = (CFL \cdot \delta x)_{max} = \frac{CFL_i}{\beta}\delta x = \alpha CFL_i \delta x = \alpha |\mathbf{U}|\delta t \tag{3.27}$$

where $\alpha = 1/\beta$. Thus, the α is dynamically computed by the model from the local and the global maximum CFL number, which varies time and space. This is done to correlate the temporal scales of the vortexes and the spatial resolution of the grid in the filter operation and it is particularly important in engine flows, where cell size and flow conditions significantly vary between the near valve region and the other regions of the mesh, and in particular from time step to time step when moving mesh is involved.

Hence the trend of filter function is that of Fig.3.3

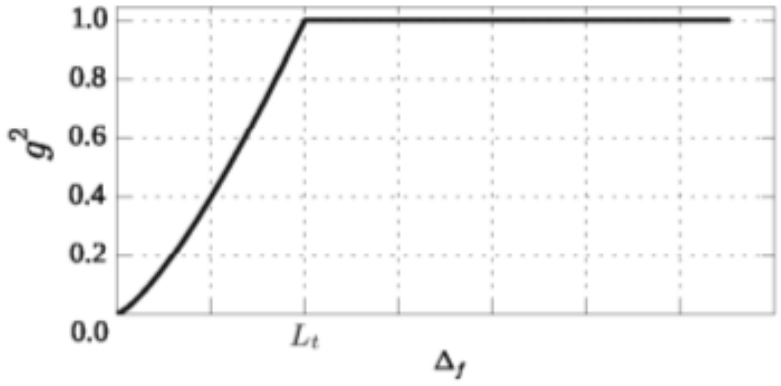


Figure 3.3. The filter function $g^2(\Delta f)$ is clipped to 1 as Δf is equal to the integral length of the modeled scales L_t and it tends to zero with a minimum as the grid size tends to the fine grid limit (implicit LES).

and in the fine grid limit, small variations of the grid size MUST correspond to high variations of the resolved scales. This behavior is justified by the choice of applying the filter not only to turbulent kinetic energy but to turbulent length scales as well. If filter was only applied to turbulent kinetic energy, filter will still tend to zero if Δf tends to zero but $\frac{\partial g^2}{\partial \Delta f}$ will still not tend to zero but it would tend to an infinite value thus its second derivative will be negative, showing a trend opposite of Fig. 3.3

3.2 Solver

3.2.1 coldTopoEngineFoam

The 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 improve convergence with multiple attaching/detaching regions and choked flows. The solver undergoes some modifications from the original development in [9] to [12] which deals with the way the ϕ_m is computed. The fundamental equations governing compressible flow inside a moving domain [13] are those shown in Sec. 2.1. In OpenFOAM mesh motion algorithm and solvers are separated: the structure of the solver is independent from the mesh motion algorithm. Thus the interface between the dynamic mesh class and the Navier-Stokes solver is minimal, being limited to a single function call that performs all mesh changes when required. In OpenFOAM®, the base transient solver for compressible viscous flows is based on a merged PISO-SIMPLE algorithm (PIMPLE) shown in Fig.3.4

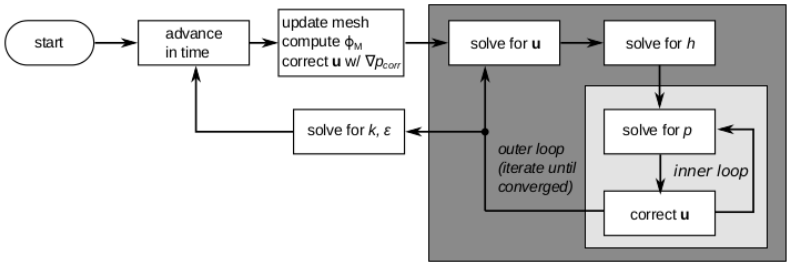


Figure 3.4. standard PIMPLE algorithm

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 time step, 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. Finally, a remapping of the newly calculated quantities is performed, the velocity correction equation 3.32 is solved and the iteration for the solution of the governing equations can start. According to the original formulation of the PIMPLE algorithm (Fig.3.4), the inner loop is rarely executed more than once (in transient-SIMPLE mode), since the outer loop allows for achievement of pressure-velocity coupling. Hence CFL can be higher than 1 because we are not resolving in PISO mode, but cannot assume too large value if we want consecutive time steps in which computed flow does not differ too much from previous time step to the following one. Thus typically has been adopted a $CFL_{max} = 5$ which allows for have a time step bigger than in PISO mode, even if actually local mesh size near wall and high values of velocity near valves region tends to limit Δt to very small values. Also 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 occurs, the energy equation is now solved together with mass conservation into the inner loop as shown in Fig.3.5

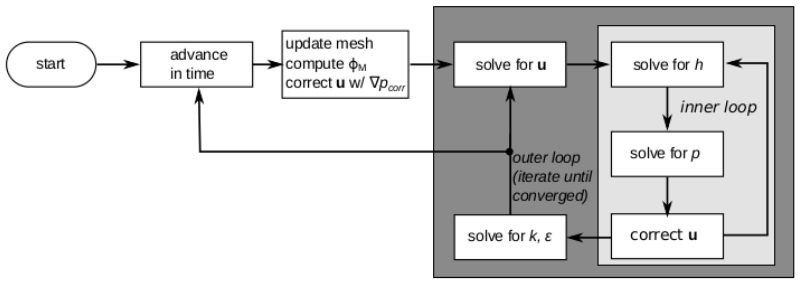


Figure 3.5. modified PIMPLE algorithm

, 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 time steps or during the attach/detach of multiple mesh regions (e.g open- 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. Moreover the stronger coupling between energy and pressure and allows for an high under-relaxation factors (up to 0.9) set-up, thus limiting the apparent overhead due to an increased number of inner iterations. The general solution procedure for the solver is as follows:

1. update time step 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 transient Simple algorithm:
   - compute mass fluxes at cell faces;
   - define and solve pressure equation (repeat multiple times for non-orthogonal mesh corrector steps);
   - correct fluxes;
   - correct velocities and apply BCs;
   - repeat for number of PISO corrector steps (in transient Simple there is only one PISO loop);
6. compute turbulence and correct velocities;
7. repeat from 1 for next time step.

Despite the formulation with moving boundaries looks very similar to the formulation with a non-moving domain [1], solution of equations 2.17, 2.18, 2.19 requires particular care because of the term including the relative advection velocity $\mathbf{u} - \mathbf{u}_b$. In fact, when they are discretized in a FV framework, advection velocities are substituted by cell face fluxes ϕ; similarly, boundary velocities $\mathbf{u}_b$ are replaced by cell face fluxes originated by points motion, ϕ_M . As shown in [13] mass source can appear in the

mass conservation equation as cell faces move, even if mesh fluxes are inserted in the discretized equations:

$$\Delta\dot{m} = \rho\frac{\Delta V}{\delta t} \tag{3.28}$$

To avoid this spurious term Eq. 3.28 one must guarantee that the *Space Conservation Law* (SCL) is fulfilled [14]. SCL can be regarded as a continuity equation in case of a zero fluid velocity:

$$\frac{d}{dt}\int_V dV - \int_S \mathbf{U}_b \cdot \mathbf{n} dS = 0 \tag{3.29}$$

Discretization of Eq. 3.29 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 $\dot{\mathbf{V}}_b$; in the simplest case of Euler implicit integration, the mesh motion flux can be calculated as:

$$\phi_M = (\mathbf{U}_b \cdot \mathbf{n})_f S_f = \dot{V}_f \tag{3.30}$$

where $\dot{V}_f = \frac{\delta V}{\Delta t}$ is s 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^0 :

$$\dot{V}_f = f(T - T^0) \tag{3.31}$$

Since a face in OpenFOAM ® is stored as a list of point IDs, and not as a list of point coordinates (Sec. 4.1.1) Eq. 3.31 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). However, when topological changes are triggered, points are renumbered and hence there is no correspondence between old and new point IDs, so the correlation between T and T^0 is no longer valid.

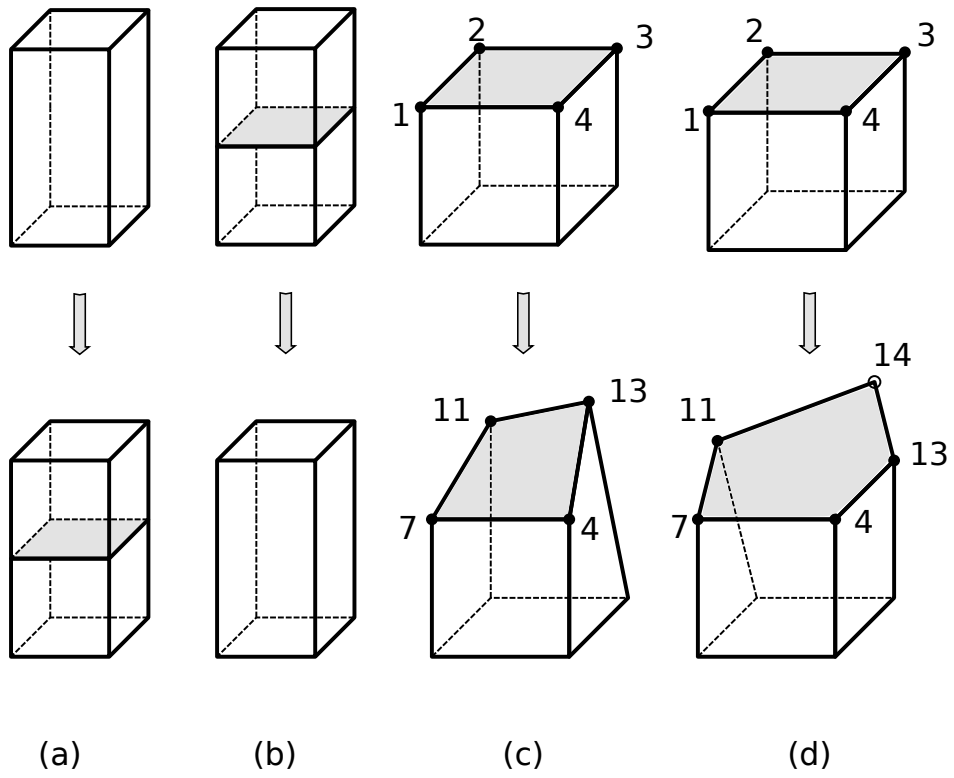


Figure 3.6. Different cases of topological changes. a) Face insertion; b) Face removal; c) Face does not topologically change, but its vertices get renumbered; d) Face is transformed and a new vertex is inserted.

Thus 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). it is useful distinguish between addition and removal of a face. If a face is added during a topology modification (Fig.3.6-a), its mesh flux must be zero. This is easily ensured by explicitly setting the value of ϕ_M on newly created faces. If a face is removed (Fig.3.6-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 will be further explained .

- a cell is modified by point addition/removal thus points are renumbered as a consequence of a topological change, but the owning faces do not show any substantial modification (Fig.3.6-c). In this case Eq. 3.31 can be still applied but face triangle decomposition T^0 must be rewritten using the new point IDs, that are deduced using a point-to-point map generated during the topological change.

- a cell simply changes its shape and not its definition, but points are added or

removed, (Fig.3.6-d). In this latter 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 as shown in Fig. 3.7. '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.

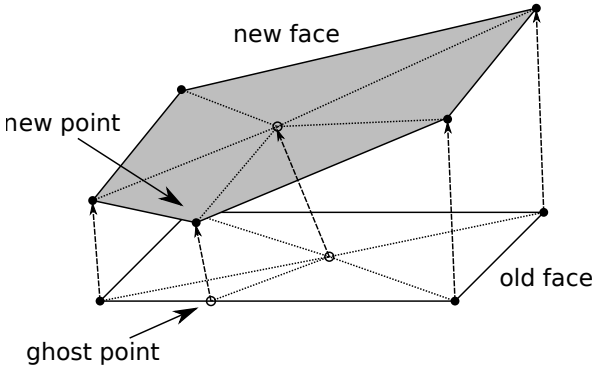


Figure 3.7. Handling of faces with inserted points. The new point is projected onto the counterpart edge originating a 'ghost' point, and the edge is split. Now both faces can be decomposed in the same number of triangles. In case the new face has less point than the old one, the ghost point is added on the new face instead.

Finally, before solving Eqs. 2.17 and 2.18 on the updated mesh, old values of $\mathbf{U}$, p and ρ must still satisfy continuity when they are remapped onto the new grid: the old velocity field ${\mathbf{u}_n}^n = \mathbf{u}(\mathbf{x}^n, t^n)$ might not be compliant with the continuity equation (Eq. 2.17), when it is re-sampled onto the new mesh. Therefore, a modified form of Poisson equation (Eq.3.32) need to be solved for a pressure corrector p corr ($p^{'}$):

$$\frac{1}{\psi}\frac{\partial p^{'}}{\partial t} + \nabla^2 p^{'} = [\nabla \cdot \phi^{n-1}]_n - \nabla \cdot [\rho \vec{U}_{f_n}^{n-1} \cdot \vec{S}_f^n] \tag{3.32}$$

where ^{n} means "at the current time step" while $_n$ means "on the mesh at the current time step". Hence :

$$[\nabla \cdot \phi^{n-1}]_n \tag{3.33}$$

represent the divergence of $\phi = \rho U$. ρU is defined at the previous time step, while its divergence is mapped on the mesh of current time step. This scalar is defined on all cell centers. while:

$$\nabla \cdot [\rho \vec{U}_{f_n}^{n-1} \cdot \vec{S}_f^n] \tag{3.34}$$

represent the flux at the previous time step on cell faces remapped on the mesh at the current time step. This scalar is not defined on all cell faces because when a topology change occurs e.g faces can be added and it is impossible to assign to it a flux. Thus the difference between 3.33 and 3.34 will be high only on cells where a topology change occurs. In this case the continuity will not be respect and a $p^{'}$ will be calculated in order to makes null the difference between the two terms. Finally the flux at the current time step on the mesh at the current time step will be:

$$\phi_n^n = \nabla \cdot [\rho \vec{U}_{f_n}^{n-1} \cdot \vec{S}_f^n] + \frac{1}{A_p} \nabla p^{'} \tag{3.35}$$

The equation 3.32 must be completed with appropriate boundary conditions. On solid walls they have to be of Neumann type ($\partial p^{'}/\partial n = 0$), whereas on permeable walls a Dirichlet boundary condition is applied ($p^{'} = 0$). The pressure correction problem assumes therefore the following form:

$$\begin{cases} \frac{1}{\psi}\frac{\partial p^{'}}{\partial t} + \nabla^2 p^{'} = [\nabla \cdot \phi^{n-1}]_n - \nabla \cdot [\rho \vec{U}_{f_n}^{n-1} \cdot \vec{S}_f^n] \\ (\partial p^{'}/\partial n = 0) \end{cases} \tag{3.36}$$

During intake and exhaust strokes there is at least one open boundary, thus Eq. 3.36 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. 3.36 is solved separately for each sub-domain (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 $p^{'}$ is imposed at an arbitrary location of the domain:

$$\begin{cases} \frac{1}{\psi}\frac{\partial p^{'}}{\partial t} + \nabla^2 p^{'} = [\nabla \cdot \phi^{n-1}]_n - \nabla \cdot [\rho \vec{U}_{f_n}^{n-1} \cdot \vec{S}_f^n] \\ (\partial p^{'}/\partial n = 0) \\ p^{'}(\mathbf{x}_0) = 0 \end{cases} \tag{3.37}$$

Once $\nabla p^{'}$ is updated Eqs. 3.32 and 3.35 are solved iteratively any time the mesh changes, until convergence on pressure is reached.

Chapter 4

Mesh generation for ICE

After a brief description of what is a mesh and which are the most common mesh quality metrics, it will be shown the dynamic mesh handling with topological changes of ICE in OpenFOAM. Then it will be described the top-down strategy technique in order to realize hexa unstructured mesh in ICEM, for each case study, according to the logic of dynamic mesh handling. Finally it will be described step by step the meshing conversion and manipulation tool used in OpenFOAM to obtain the final mesh on which the simulation is done.

4.1 Discretization of spatial volume and mesh quality metrics

Mesh generation consists in dividing the physical domain into a finite number of discrete regions, called control volumes or cells in which the solution is sought (domain discretization). The computational grid must be developed trying to find the compromise between grid refinement and computing time. Too refined meshes allow to achieve precise results, but the computing time may become incompatible with work. The mesh density should be high enough to capture all relevant flow features. In areas where the solution changes slowly, larger elements can be used. Generating high quality meshes is a critical step for CFD computations. Depending on the quality of the mesh, very different results can be obtained, which can make post-processing and interpretation of the solution a difficult task, due to contrasting results caused by meshing issues. No single standard benchmark or metric that can effectively assess the quality of a mesh exists, but there are suggested practices to follow. The most common mesh quality metrics are:

- Orthogonality
- Skewness

- Aspect ratio
- Smoothness

Referring to figure 4.1 mesh orthogonality is the angular deviation of the vector **S** (located at the face center f) from the vector **d** connecting the two cell centers P and N.

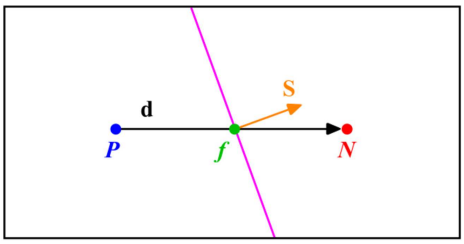


Figure 4.1. Mesh orthogonality

$$i_{face,orth} = \frac{\boldsymbol{d}\Delta}{|\boldsymbol{d}||\Delta|} \tag{4.1}$$

Mesh orthogonality affects the gradient of the face center f and it adds diffusion to the solution.

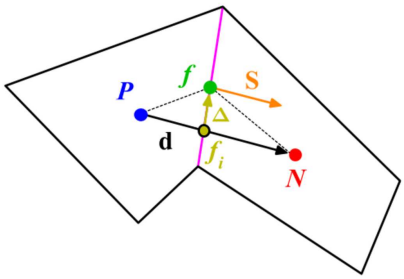


Figure 4.2. Mesh skewness

Skewness is the deviation of the vector **d** that connects the two cells P and N to the face center f. The deviation vector is represented with Δ and f_i is the point where the vector **d** intersects the face f .

$$i_{face,skewness} = \frac{\Delta}{\boldsymbol{d}} \tag{4.2}$$

Skewness affects the interpolation of the cell centered quantities to the face center f and it adds diffusion to the solution as well.

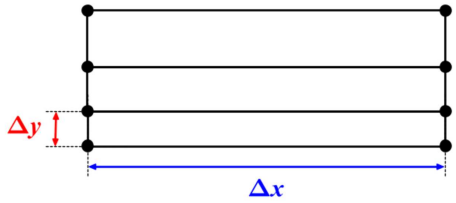


Figure 4.3. Quality metrics: Mesh aspect ratio

Mesh aspect ratio AR is the ratio between the longest side Δx and the shortest side Δy Large aspect ratio is fine if gradients in the long direction are small, but usually high aspect ratio leads to smear gradients. Smoothness, also known as expansion rate, growth factor or uniformity, defines the transition in size between contiguous cells.

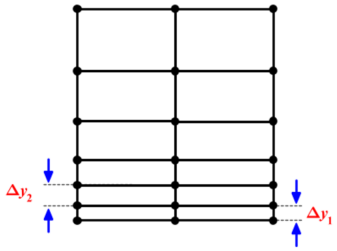


Figure 4.4. Mesh smooth transition

Large transition ratios between cells add diffusion to the solution; ideally the maximum change in mesh spacing should be less than 20%:

$$\frac{\Delta y_1}{\Delta y_2} = 1.2 \tag{4.3}$$

4.1.1 Mesh definition in OpenFOAM

In order to understand how topology modifier works is useful clarify on what domain discretization within OpenFOAM is relied on. Domain discretizazion is based on an unstructured mesh of polyhedral cells with an arbitrary number of faces. Mesh definition is enclosed in five different files contained in `PolyMesh` file under the `constant`

directory:

- `points`: contains the coordinates of all mesh nodes (vertices) in form of a list. The position index of a point into the list, starting from zero, represents its label or *addressing*. The point list cannot contain two different points at an exactly identical position nor any point that is not part at least one face.

- `faces`: contains the definition of faces in form of a list of arrays. Each face is defined as a list of points, that are identified by their respective labels; the number of point constituting a face can be arbitrarily large. Their order inside the face definition identifies the direction of face normal vector, according to the right hand rule. Again, position index of face into this file gives its addressing

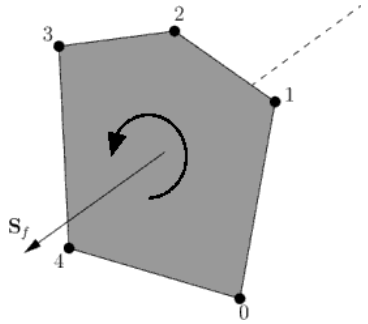


Figure 4.5. Right hand rule to define the direction of face normal vector

- `owner`: each face is *owned* by a cell, whose lebel is specified in the file *owner*. The i-th element of the list is the addressing of the cell that owns the face addressed by the position index i

- `neighbour`: the cell adjacent to face i on the opposite side with respect to the owner cell is the face texttt*neighbour*. The neighbour cells are specified in this file following the same logic used fr owners. Not all faces have a neighbour: boundary faces have only an owner; in this case their addressing will be -1.

- `boundary`: contains specification of boundary "patches" onto which the physical boundary condition for each variable will be defined. Each patch is a list of consecutive face labels, and it is defined through the start face addressing and the number of faces the patch is composed of. Boundary faces always lie at the end of the face list, after internal faces.

Splitting the mesh definition in different files allows for reducing the storage memory overhead in case of simulation with dynamic meshes: if no topology change is involved,

only the point coordinates have to be saved in the result folders, otherwise if mesh topology changes, the complete mesh definition must be stored for restart and post processing. Moreover, every time the mesh is read or updated, a strict internal checking is performed to verify self-consistence of its definition. Note that parts that can be easily inferred from mesh definition, like edges or cell topology itself, are not considered in it as well. At the same way, it is possible to specify subsets of point, faces or cells called "zones", storing the affected entities addressed in a file called respectively `pointZones`, `faceZones` , `cellZones` which can be automatically update any time the topology changes.

4.2 Dynamic mesh strategy

Dynamic mesh handling in CFD codes is a suitable technique to simulate in-cylinder flows in ICE (moving valves and piston). OpenFOAM represents a suitable platform for complex physical modelling, since it includes a common interface to all dynamically changing meshes. In order to preserve the mesh quality during extreme boundary deformation due to piston and valve motion, the number of the cells in the mesh need to be changed. This is the reason that leads to define a set of "topological changes", allowing the possibility of attaching or detaching boundaries, adding or removing cell layers and using sliding meshes interfaces. In topological changes the idea is to use self-contained objects, where a topology change is executed on demand, rather than at prescribed moments. Mesh motion can be performed in two ways, namely **layering** and **deformation**. In the first method the mesh topology is changed by adding or removing cells, lefting unchanged the position of most mesh points, while in the second method the majority of points in the cylinder is moved and the motion of internal point is obtained by solving a mesh motion equation. Boundary zones can include non-conformal interfaces, where the boundaries between cell zones present mesh node locations that are not identical and they are connected to each other by passing fluxes. Region are then disconnected when mesh points are moved. Also sliding interface has been used to reduce the mesh size in the generation of cylinder grids, characterized by a high quality mesh near valve region.

4.2.1 Topology modifier

In order to realize the topological changes, the moving boundaries are handled by a wide used of so called *topology modifier* namely:

- Sliding interface, to connect dynamically different mesh regions through non conformal interfaces;

- dynamic addition/removal of cell layers, to keep optimum size of the cells during piston and valve motion;

- attach/detach of boundaries, to automatically simulate the valve closure event;

In OpenFOAM, all topological changes, are executed by self-contained objects, called mesh modifiers, that are activated on demand whenever a topology change is requested, rather than at prescribed moments. A common interface and the use of virtual functions allows for a common top level code and run-time selection of topology changers. When the top-level application triggers a mesh update, a loop is performed over all modifiers; if a topological action is requested, the corresponding algorithm embedded in the specific modifier is invoked. As the loop proceeds, the modified mesh is stored in a temporary location of memory and an object called `mapPolyMesh` is created. Since points-,faces- and possibly cell- addressing change during a topological action, the `mapPolyMesh` object stores all correspondences between original and modified mesh. This is used to recalculate the actual FV mesh, zones, fields, mesh modifiers etc. Only after al mesh modifiers have been executed, the global FV mesh is changed and execution can continue.

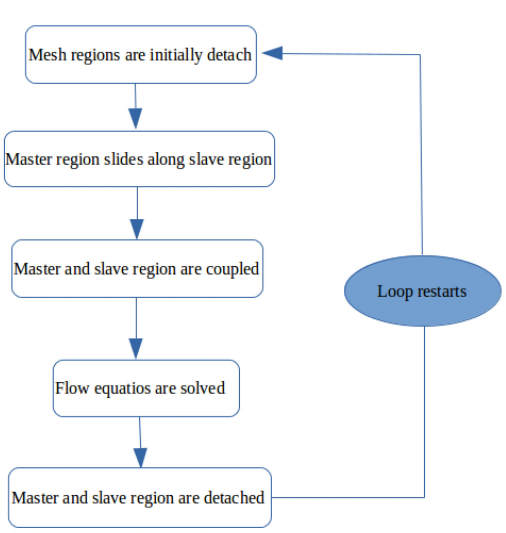


Figure 4.6. moving points algorithm

Sliding interface: coupling and decoupling algorithm

It allows for the creation of a reversible, non conformal-grid interface between two mesh regions with different mesh structure. The coupling is achieved by changing the local mesh topology in order to get strict one-to-one point correspondence. After the coupling operation, no "interface" still exist, and the former detached regions can be considered as one. Therefore, the sliding interface philosophy is different from other region coupling algorithms, that mostly rely on baffle faces to achieve a fluid-dynamic

link between topologically separated zones. If necessary, the topological change can be reversed and mesh can be brought back to the detached configuration. An example is when a sliding motion is prescribed between two different mesh regions, namely master region and slave region. The algorithm employed to move points without degrading the cell quality is made up of some steps as you can see in figure 4.6

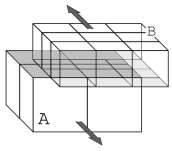


Figure 4.7. Sliding interface between master and slave regions with different mesh structure

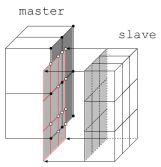


Figure 4.8. Point projection of slave patch (red) onto master. Solid dots are master retained points, hollow circles are slave points added because of direct hits or intersection between the two patches

To proceed with the operation performed by this topology changer the user is requested to specify:

1. the opposite facing surfaces which will be attached by the sliding interface at the start time. They need to be valid patches belonging to the external surface of the mesh, lying on the same ideal surface, up to a tolerance chosen by the user, and the obviously must overlap;

2. which one of the surface is the "master" patch, while the other one will be implicitly defined as "slave" patch. The difference between slave and master lies in the fact that the former will possibly undergo point deletion to adapt to other side, while the latter will remain unchanged as long as possible. Therefore all the original points on the master patch are retained and some new points, coming from the slave side, could be added. In this way the face definition changes for both sides.

When the topology modifier is triggered, the topological change proceeds automatically : points belonging to the "slave" side are projected onto the "master" patch and in case of no valid projection, the algorithm terminates without errors. Actually projected point location is corrected for direct **point-to-point**, **point-to-edge** and **edge-to-point** hit, eliminating all degenerate cases of intersection. In case of a direct point-to-point hit the slave point is merged with the master point which is retained, and the slave point is removed. In case of point-to-edge hit, the affected edge is split and the adjacent faces are modified accordingly. The case of edge-to-edge intersection

follows the same rules but now a point is added at intersection. Finally, if projected point lies onto a face, this is splitted itself by adding new edges. The process is shown in Fig. 4.8.

Then an enriched patch, containing all points and intersections of both sets of faces is assembled and used to create consistent mesh structure: faces sets are isolated using right-hand-walk algorithm. Afterward, original master and slave faces involved in the projection are removed and replaced with the corresponding ones coming form enriched patch and newly inserted faces is linked to an owner and neighbour cell. Faces involved into he projection but not master nor slave are called *stick-outs*, i.g internal faces with an edge lying on the interface. After the coupling algorithm completion, the resulting mesh will have seamless junction between the former separated region, as shown in Fig. 4.9.

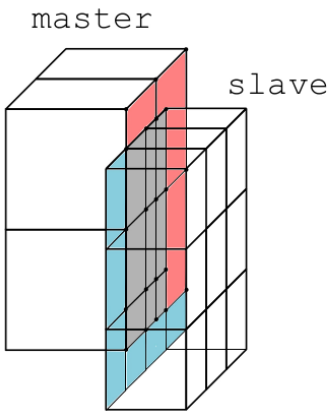
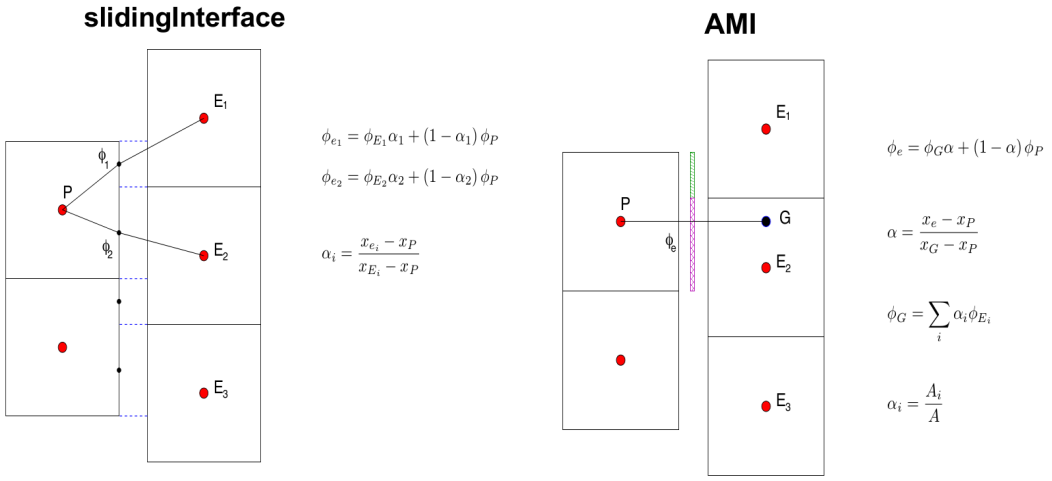


Figure 4.9. Coupled interface:red and blue faces represent what remains of the original master and slave patch after a coupling

Since the different mesh region are merged, topology modifications have to be stored in order to allows the further operation of decoupling algorithm. Thus all point positions, face, edge and cell definitions must be saved as they were before the coupling, before proceeding with topological change. In [15] the information about removed entities has been changed: faces and points belonging to the enriched patch created at the projection time are saved in appropriate *faceZones* while removed entities definitions are written to a separate *meshModifiers* file per each time step. Hence, decoupling of the interface relies completely on the information saved during the previous stage and the two interfaces are brought back to life as they were originally. After a successful decouple, no information needs to be saved but the master and slave patch labels, thus the mesh modifiers is cleared out.

Thus the sliding interface can be used to create a link between mesh regions with non-conformal interfaces, as done in [9]. However, if compared to the supermesh approach (AMI), the target mesh approach (slidingInterface) is more demanding in terms of computational resources, because coupling/decoupling algotirthm updates the mesh topology at any time step. For this reason in [12] Arbitrary Mesh Interface approach is used to move valves. The latter is a supermesh approach which relies on vertex-based solution using a bounded Galerkin projection over a virtual triangulated surface mesh. Moreover it allows to avoid updates in mesh change every time step. Although AMI is faster and has an optimized solution with full overlap of fluid regions it is not as stable as *slidingInterface* with partial overlap of mesh regions. The difference between the method of calculation of fluxes of two approach are shown in Fig.4.10.



(a) *Calculation of fluxes with SlidingInterface.* (b) *calculation of fluxes with AMI.*

Figure 4.10. Comparison between target mesh approach and supermesh approach

while in Fig.4.10a is shown a typical Second order interpolation of face-fluxes over the interface, after the coupling algorithm, in Fig. 4.10b is shown a weighting of the cells sharing part of their surface with the same face on the coupled patch to calculate the fluxes. For these reason in Sec.4.3.2 AMI has been used on slidinIntake and slidingExhaust non-conformal patches, while slidingInterface has been applied between cylinder and spark plug (non-conformal static interface) through `stitchAndSplitMesh` tool, thus only one coupling occurs between them. That means that cylinder region and intake/exhuast region will be always belong to different regions during simulation while spark plug region becomes part of cylinder region.

Layer additional/removal

It consists in adding or removing layers of cells in accordance with the motion of a moving boundary (e.g the piston patch as in Sec.4.3.2 and Sec. 4.3.1) whereas the majority of the grids remain fixed. This strategy allows for keeping the mesh quality constant during the whole simulation: cells are added when the thickness of the stretched layer rises above a specified threshold and they are removed when the deforming layer thickness falls below a different threshold. In both axis-centerd valve and TCC case, additional/removal of cell layer has been applied to piston motion defining in `engineGeometry` file:

- the name of the patch;
- the cellSet to move and the faceSet on which layer A/R must be applied; the generation of these will be explained in Sec.4.4.6.
- The parameters to control the maximum and the minimum thickness of the cell layers to add or remove.

While layer A/R has been applied to intake and exhaust valve in TCC case, thus in `engineGeometry` file has been also defined faceSet and cellSet of the two valves.

Attach/detach of boundaries

Attach/detach mesh modifiers 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 this work all internal faceSet has been created directly with ICEM, which has been used to generate the hexaedral mesh in both cases. Thus the *attachDetach* C++ class separates the intake/exhaust ports from the combustion chamber at valve closure.

4.3 Mesh generation strategy with ANSYS ICEM CFD

The main aim of the work is the creation of a high quality unstructured hexaedrical mesh in order to obtain a particular kind of mesh shape that is coherent with the turbulence model either with the topological changes involved in ICE motion. The former purpose provides for the creation of layers with increasing spacing near wall patches and the number of cells near jets as well. The latter is obtained building the block, on which the mesh is based on, according to the type of motion the cell set or the face set are involved in and according to the kind of topology modifier to use. For these reason i decide to use ANSY ICEM CFD.

Introduction to ANSYS ICEM CFD

ANSYS ICEM CFD provides advanced geometry acquisition, mesh generation, and mesh optimization tools to meet the requirement for integrated mesh generation for sophisticated analyses. Maintaining a close relationship with the geometry during mesh generation, ANSYS ICEM CFD's mesh generation tools offer the capability to parametrically create meshes from geometry in numerous formats:

- Multiblock structured
- Unstructured hexahedral
- Unstructured tetrahedral
- Hybrid meshes comprising hexahedral, tetrahedral, pyramidal and/or prismatic elements

Beginning with a robust geometry module which supports the creation and modification of surfaces, curves and points, ANSYS ICEM CFD's open geometry database offers the flexibility to combine geometric information in various formats for mesh generation. The resulting structured or unstructured meshes, topology, inter-domain connectivity and boundary conditions are then stored in a database where they can easily be translated to input files formatted for a particular solver, in this case OpenFOAM, which accepts only unstructured mesh. In order to build the grid it is used a hexa meshing modules because it allows to obtain an oriented grid within the domain either good mesh quality metrics which will not afflict too much the simulation with numerical errors. The ANSYS ICEM CFD Hexa mesher is a semi-automated meshing module which allows rapid generation of multi-block structured or unstructured hexahedral volume meshes representing a new approach to grid generation where the operations most often performed by experts are automated and made available at the touch of a button. Blocks can be built and interactively adjusted to the underlying CAD geometry. This blocking can be used as a template for other similar geometries for full parametric capabilities. Complex topologies, such as internal or external O-grids can also be generated automatically.

The basic difference between structured and unstructured grids lies in the form of the data structure which most appropriately describes the grid. A structured grid of quadrilaterals consists of a set of coordinates and connectivities that naturally map into elements of a matrix. Neighboring points in a mesh in the physical space are the neighboring elements in the mesh matrix.

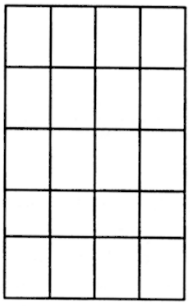


Figure 4.11. structured grid

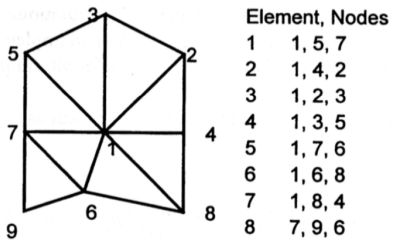
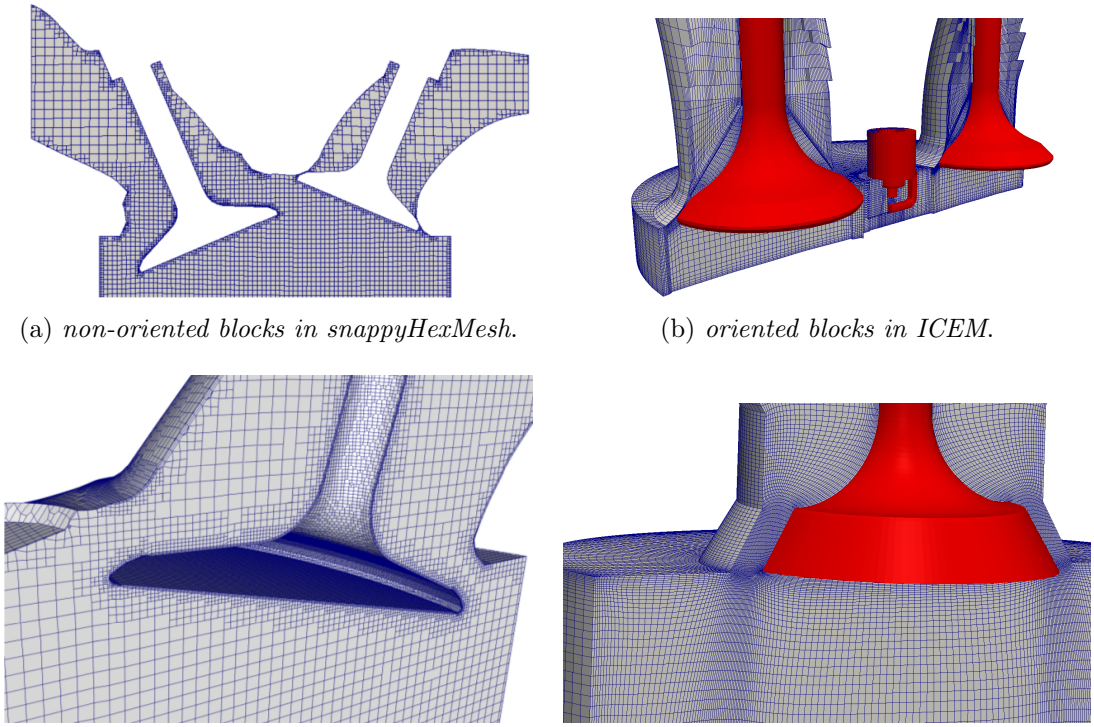


Figure 4.12. unstructured grid

Thus, for example, a two-dimensional array $x(i, j)$ can be used to store the x-coordinates of points in a 2D grid. The index i can be chosen to describe the position of points in one direction, while j describes the position of points in the other direction. Hence, in this way, the indices i and j represent the two families of curvilinear lines. These ideas naturally extend to three dimensions. For an unstructured mesh the points cannot be represented in such a manner and additional information has to be provided. For any particular point, the connection with other points must be defined explicitly in the connectivity matrix, in OpenFOAM represented by a list of coordinates, points or faces if we are considering points, faces or cells respectively Fig. 4.12).

Thus the choice of build an unstructured mesh depends especially on simulating software. Anyway the unstructured mesh has a real advantage because the points and connectivities do not possess any global structure. It is possible, therefore, to add and delete nodes and elements as the geometry requires or, in a flow adaptivity scheme, as flow gradients or errors evolve. Hence the unstructured approach is ideally suited for the discretization of complicated geometrical domains and complex flow field features as internal combustion engine or injectors. However, the lack of any global directional features in an unstructured grid makes the application of line sweep solution algorithms more difficult to apply than on structured grids.

Although the choice of using OpenFOAM, which arises by the use of the dynamic mesh tools developed custom-made for this software, i am not going to use its mesher `snappyHexMesh` but a commercial one, ANSYS ICEM CFD. This decision rely on some features of the mesher that makes the latter better than the former for the purpose of the work:
The snappyHexMesh utility generates 3-dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) automatically from triangulated surface geometries in STereolithography (STL) format, however it is more difficult to obtain oriented blocks with geometry with internal baffles (here for valve motion), as shown in Figs. 4.13a, 4.13b, where topology modifier will be applied.



(a) *non-oriented blocks in snappyHexMesh.*

(b) *oriented blocks in ICEM.*

(c) *layers around valve in snappyHexMesh.*

(d) *layers around valve in ICEM.*

Figure 4.13. comparison between snappyHexMesh and ANSYS ICEM CFD around a valve in ICE

4.3.1 Flat-top cylinder head with a fixed, axis-centered valve

The axial-symmetric piston-valve assembly 4.14 introduced by Morse, Whitelaw, and Yianneskis [16] has been selected as a first test-case. In the recent years, many LES studies have been done: Haworth [17] tested on the same engine different configuration of subgrid scale models and resolutions and the influence of model constants, integration time step and mesh resolution; Mittala et al. [18] performed LES simulation by a finite difference based structured methodology, where the motion of the valves and piston was handled using a dynamic cell blanking approach and the Arbitrary Lagrangian Eulerian (ALE) method; F. Piscaglia and Onorati [19] studied on the the same simplified geometry the Complex unsteady features of turbulent fields like laminar-to-turbulent transition and tumble vortexes evolution in order to prove that the proposed approach is reliable for reproducing the dynamic behavior of complex turbulent structures in IC engines.

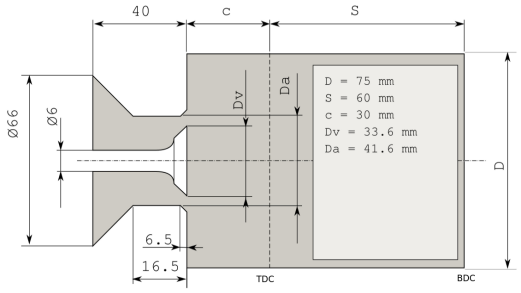


Figure 4.14. Geometry of the experimental apparatus used by Morse, Whitelaw, and Yianneskis [16]. Legend: c=clearance; S=stroke; D=bore. Lengths are in mm

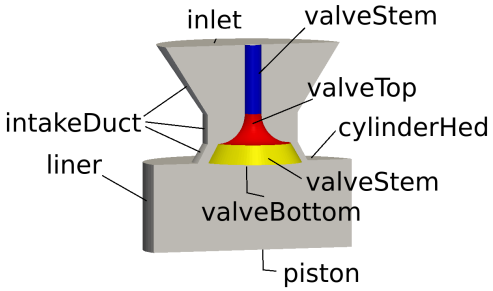


Figure 4.15. Definition of patches for mesh generation and simulation

In this particular case the only part involved in order to provide the change of topology are:

- piston, whose kind of motion have been defined in the `engineGeometry` file in OpenFOAM under `pistonMotionType`;
- liner, whose length increase or decrease according to piston motion;

Geometry definition

An accurate solution reflects the underlying geometry. The higher is the quality of geometry file imported in ICEM the higher is the mesh we will expect. To obtain such things, ICEM CFD provides:

- **Geometry import**,directly from CAD package, third party formats e.g STEP, IGES and via Workbench or Design Modeler;
- **Surface geometry kernel** which is just the part of the code that handles its native geometry. ICEM CFD's geometry kernel supports two primary geometry types, faceted and bspline;
- **Internal CAD tools** such as creation of geometry, geometry modification and repairing;

The meshing procedure starts from the import of the case geometry in ICEM and afterwards goes on with a geometry modification. To import a geometry in ICEM it is possible choose among 3 group:

1. faceted. Faceted geometry is made up of lots of little triangles. This data can come from scanners or some low end geometry tools. The common one is STereoLithography file format which describes a raw unstructured triangulated surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system. However the STL often give a rough interpretation of the surface as in Fig. 4.16.

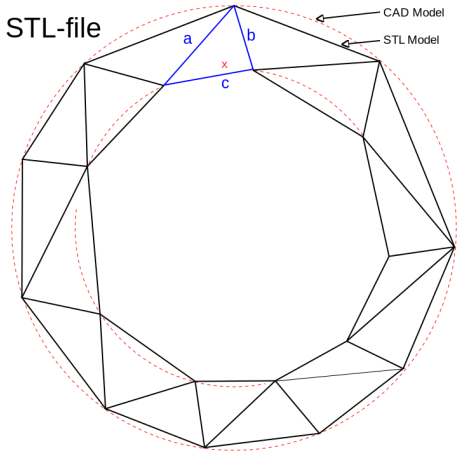


Figure 4.16. STL representation versus a CAD one, of a simple surface

2. Legacy. This comes from higher end cad tools produce basis spline (B-spline) geometry which is much higher quality because rely upon a very good mathematical definition which gives it numerical robustness and thus also a geometrical one. IGES and STEP file belong to this group.

3. Formatted data point. It allows auto curve/creation from a table of regular points;

Since STL roughness leads necessarily to a huge modification of geometry in ICEM, due to the possible presence of holes and edge misalignment that affect mesh quality, the meshing approach presented in this work uses a IGES geometry file.

The following case, whose geometry is represented in Fig. 4.17

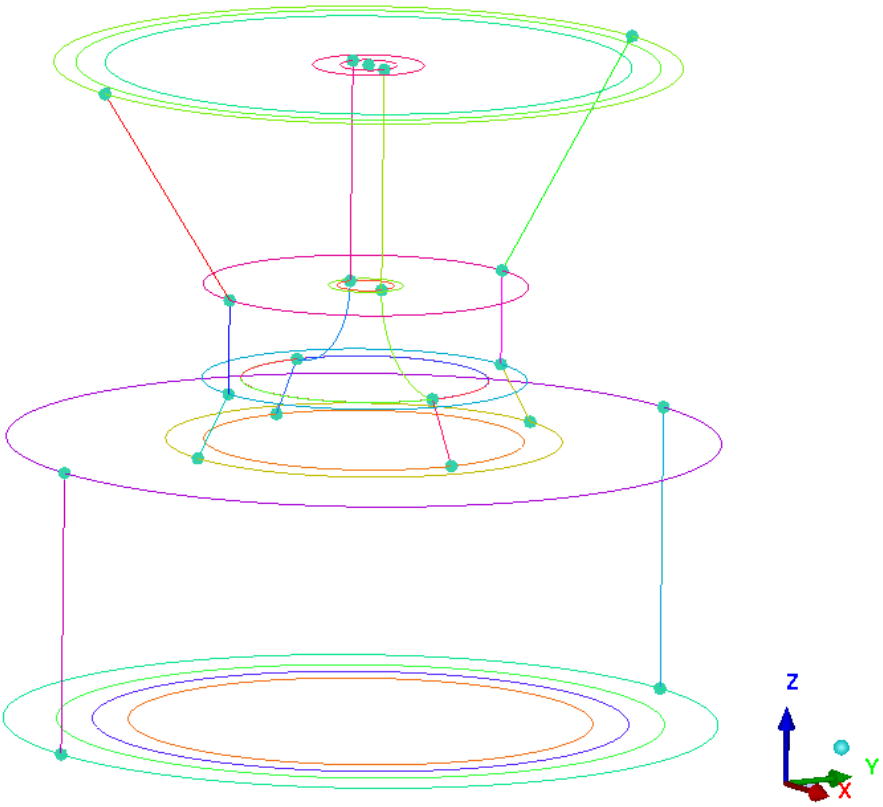


Figure 4.17. geometry representation in ICEM

Show the final geometry after the application of the repair tool which allows, up to a specified tolerance, to obtain points from each curves intersection and the the right definition of each curve. Afterward additional curves have been created in order to fix the edges of the block in a prescribed zone as It will show in blocking creation.

Blocking definition

The next step is the creation of block structure. Due to the top-down approach, the mesh will be generate from the volume. Thus using blocks there is an advantage in creating easily internal boundary or mesh zone were it is necessary to have a specific orientation of cells instead of a Cartesian one. The creation start from the curves which identify the liner, represented in black in Fig. 4.18

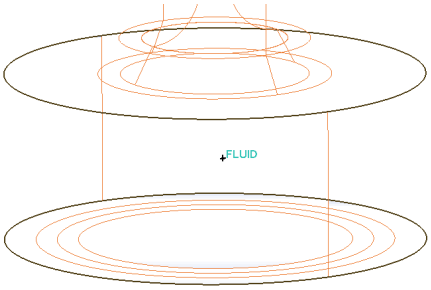


Figure 4.18. in black, curves which identifies liner

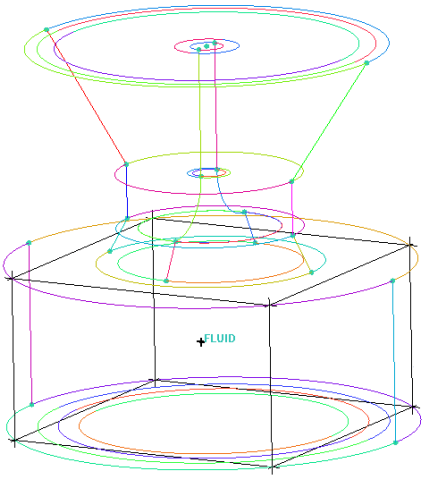


Figure 4.19. Black edges identify the block for the liner

ICEM creates the 3D block automatically around the geometry based on selected curves. Now the block has not been associated to the geometrical surface yet. In order to obtain a complete association edges and faces belonging to the block have to be associated respectively to curves and surfaces belonging to the geometry through the "**associate block to geometry**" command. This procedure will have to be repeated for each edge and face belonging to a boundary patch. It is worth made it each time a block is created, if at least one blocks' face or edge belong to a boundary patch. It is possible to do the same thing with blocks' vertexes associating them to points of geometry. The result of association of block onto geometry is represented by:

- green edge if it is associated with a curve, otherwise it is black if associated with surface. Light-blue if it is an internal edge (Fig. 4.20).

- Red vertex if it associated with a point, otherwise it is black if associated with surface. Light-blue if it is an internal vertex.

- Faces do not have such kind of association control. Default option in ICEM does not provide faces display. Thus after activating face projection it will be possible to see the face association to the boundary patch or internal baffle (Fig. 4.21).

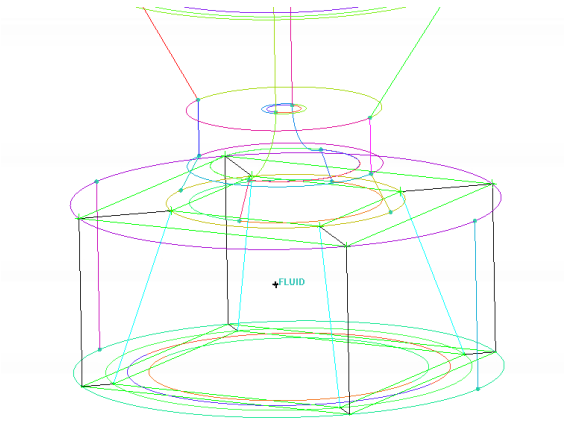


Figure 4.20. Representation of edges association: green (associated with curve), black (associated with surface), light-blue (internal edges)

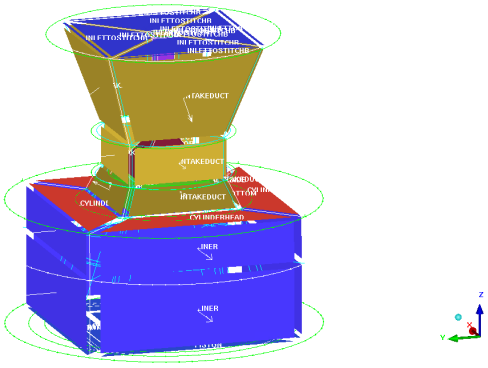


Figure 4.21. Face association with boundary patch whose label name is shown

As it can possible see in Fig. 4.20 an hexaedral block within another one. This kind of block structure is named O-Grid. It is necessary In order to create a mesh within a cylindrical geometry, thus it will be developed along all the height of the geometry. This structure it' s automatically created by ICEM . The user has only to choose the block on which O-grid will be made on, and faces on which the faces belonging to the new internal block have to be coincident. It is also possible to choose the ratio between internal and external block.

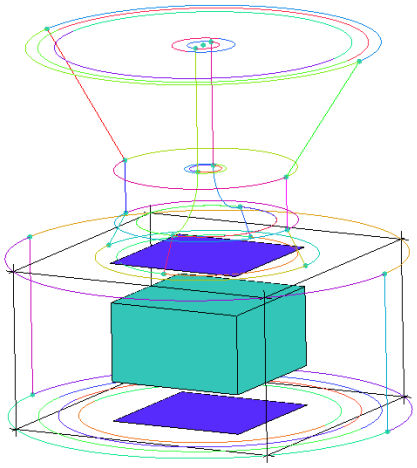


Figure 4.22. O-grid construction: selected faces in blue whie in light-blue selected block.

O-grid ensures the generation of a mesh with good quality near wall and generation of boundary layer. The difference between hexa and O-Grid mapped onto cylinder is very remarkable.

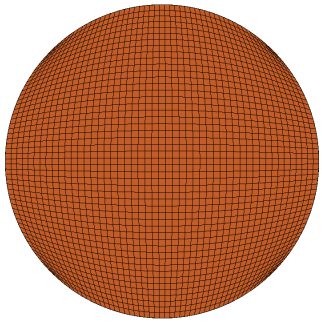


Figure 4.23. Hexaedral block mapped onto a cylindrical geometry.

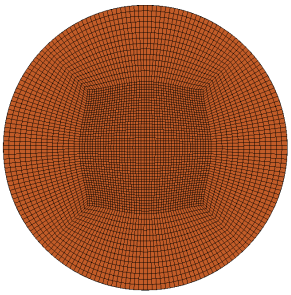


Figure 4.24. O-grid block

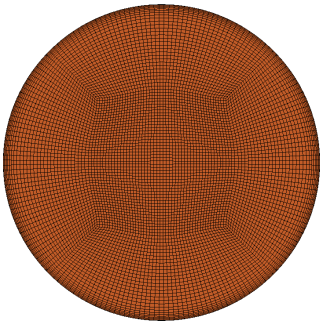


Figure 4.25. O-Grid quality near wall.

In Fig. 4.23 it is shown an hexaedral block mapped onto a cylindrical geometry. Cells near wall have a very bad quality, which reflects upon numerical convergence, and does not allow a good representation of boundary layer especially of the velocity gradient near wall. In Fig. 4.24 it is shown mesh originates from O-grid. It does not present the aforementioned problem. Moreover in order to calculate a velocity gradient near wall enough close to the real one a smooth layer thickness grading has to be created from wall to internal field. This feature can be obtain:

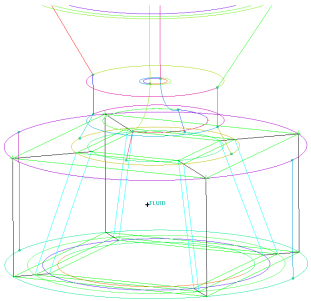
- changing spacing ratio;
- increasing number of nodes on the edge which link external and internal hexaedral-block;
- changing interpolation law on the edge.

The result of these operations are visible in figure 4.25.

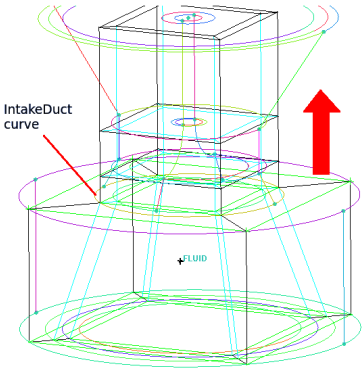
Compared with Fig. 4.24, it shows smooth transition from wall to the center and good layering near wall. It is important underlying that even if O-grid is automatically generated in ICEM representing a powerful tool, it is not easy applicable to complex geometry: the user have to choose carefully on which of several blocks to apply it and how many times he has to apply in order to have the boundary layer and oriented mesh wanted.

In Flat-top cylinder case, the generation of the mesh for valve boundary patches (Fig. 4.26c) is taken into account, making a second O-grid (Fig.4.26a) in the previous

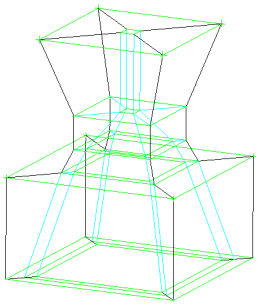
one and afterward extruding the faces (Fig. 4.26b) within the region delimited by intakeDuct diameter.



(a) *Creation of the second O-grid in the region identified by the liner.*



(b) *Extrusion of faces within the region delimited by intakeDuct diameter .*

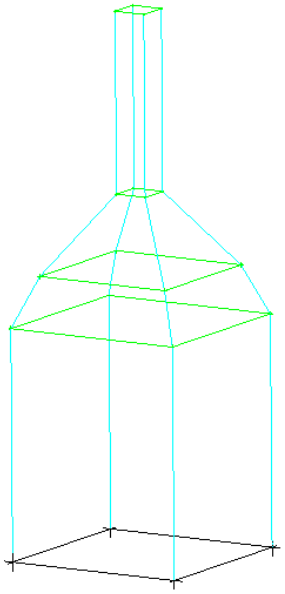


(c) *O-grid around valve geometry.*

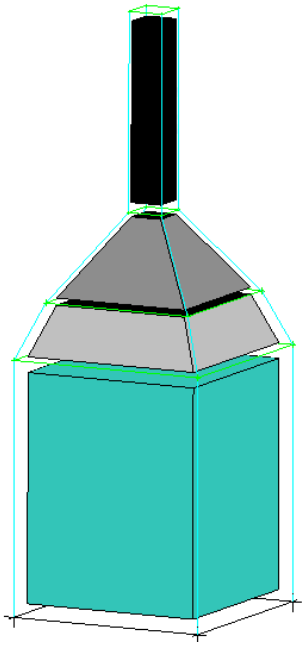
Figure 4.26. Steps for the creation of initial block around valve with all association

Figure 4.26c shows that block representing valve are internal block since valve is an internal part of geometry, but its edges are light-blue, thus edges are not associated with valve geometry. In order to do that, blocks representing valve has to be deleted, afterwards external faces of block already deleted will be associated with boundary patch belonging to valve.

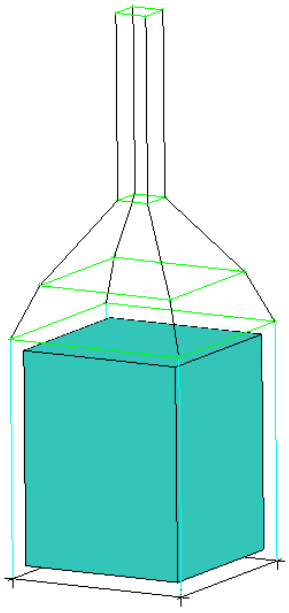
Carrying on with blocking procedure difficulties on visualizations occurs. In order to delete blocks with ease, ICEM allows to isolate internal blocks (Fig. 4.27a). After having deleted the three block (Fig. 4.27c), edges will became black due to association with underlying valve geometry (Fig. 4.27d).



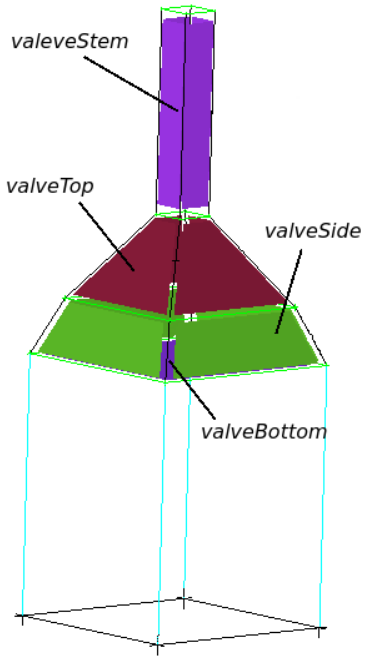
(a) *Isolation of the most internal block. Edges are light-blue.*



(b) *Selection of three blocks representing valve.*



(c) *Valve block are deleted and edge has became black and the three block selected previously there are not any more.*



(d) *Association of faces to geometry surface.*

Figure 4.27. steps for valve-block deletion

The result of these step is the same of Fig. 4.26c but now there are no more block inside where the valve resides. Although two O-grid have been already created, to obtain a good mesh , especially near valve patches, two more O-grid have to be created:

1. The first needs to guarantee a boundary layer around valve and along intakeDuct (Figs.4.28,4.29);

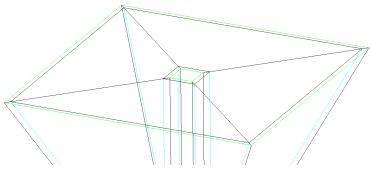


Figure 4.28. O-grid for intakeDuct and valve-stem

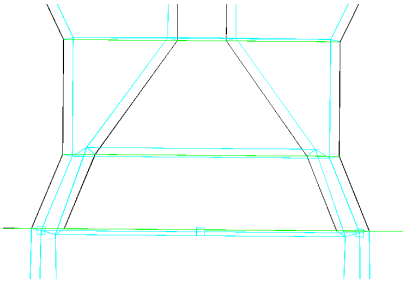


Figure 4.29. O-grid for valve

2. The second needs to provide the typical structure of O-grid under the valveBottom patch as in Fig. 4.30;

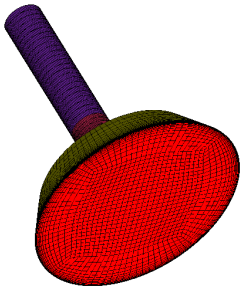
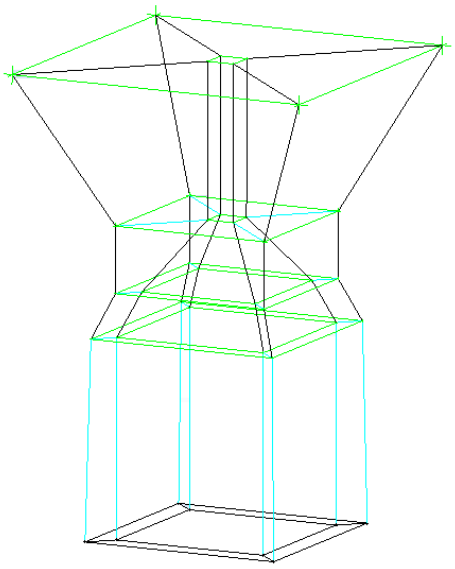
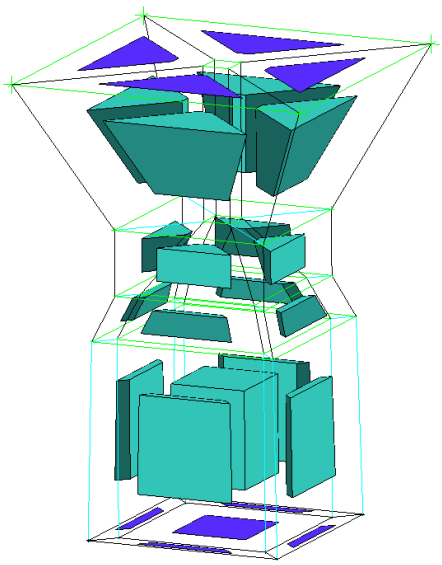


Figure 4.30. valve-bottom O-grid

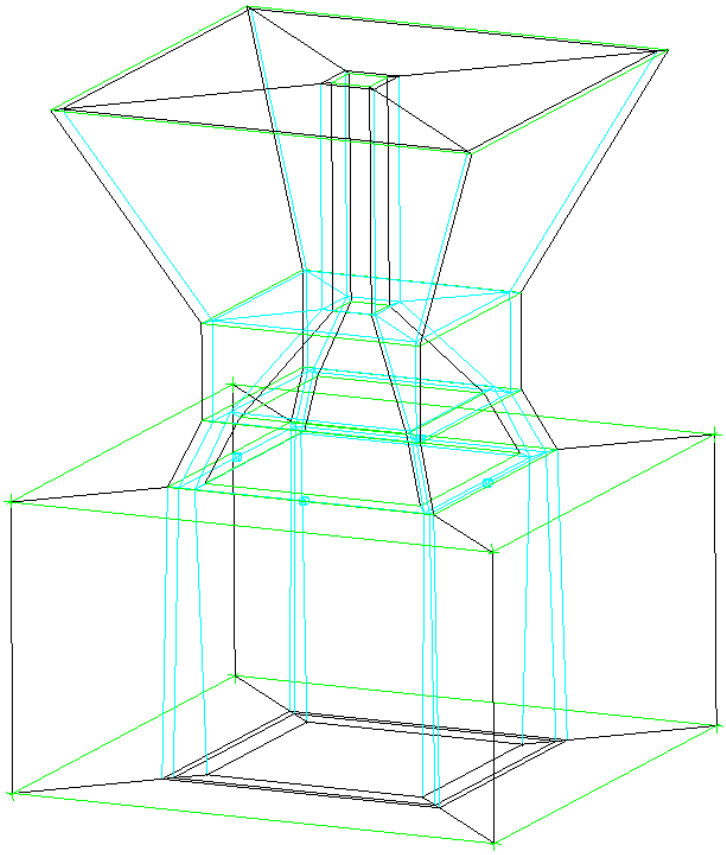
The former target is reached isolating the blocks surrounding the valve (Fig. 4.31a). Afterwards the selection in this case is simple due to the fact that all blocks will be involved in, thus all visible blocks can be selected, while only top and bottom faces must be selected to obtain the final structure in Fig. 4.31c. The latter target follow the same step on the block under the valveBottom patch (Fig. 4.32)



(a) *Blocks isolation.*

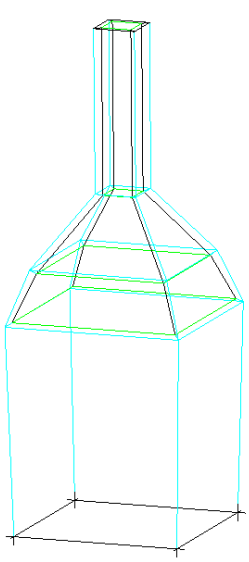


(b) *Blocks and faces selection .*

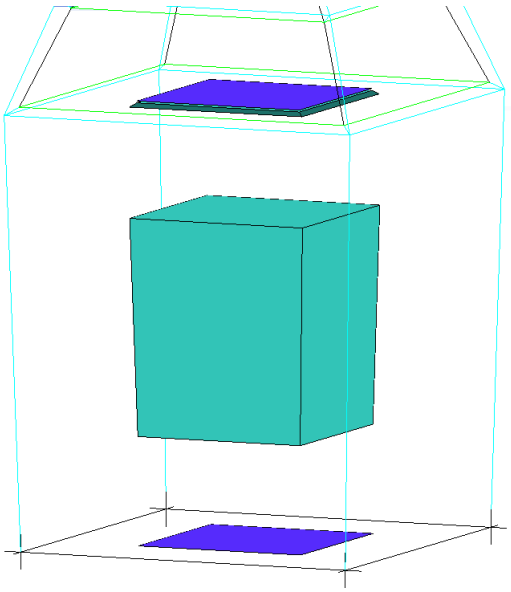


(c) *Final structure.*

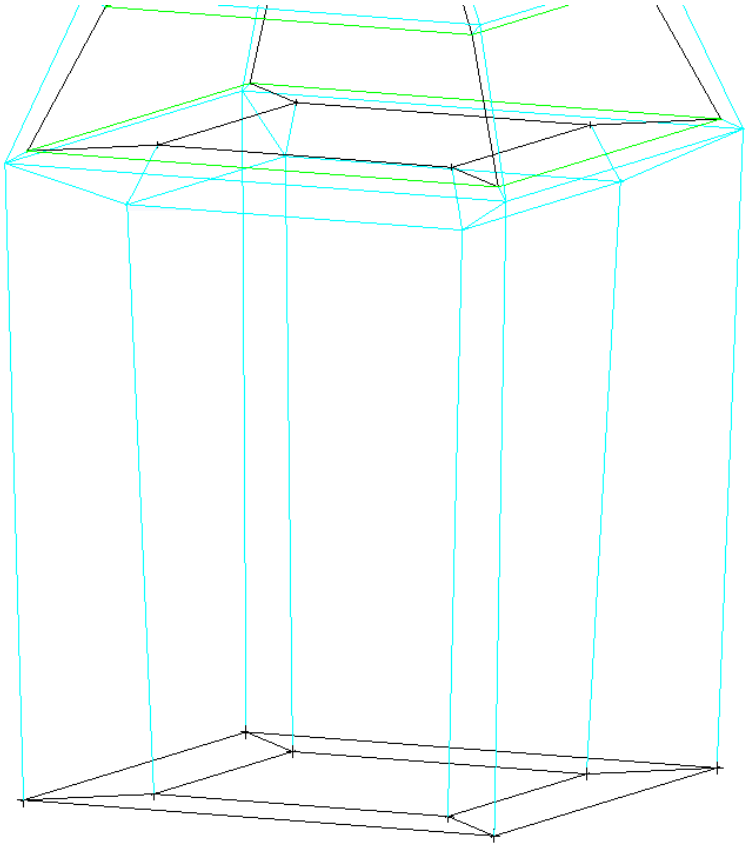
Figure 4.31. In figure are presented the three steps to obtain a boundary layer along intakeDuct and valve patches



(a) *Blocks isolation.*



(b) *Blocks and faces selection .*



(c) *Final structure.*

Figure 4.32. In figure are presented the three steps to obtain the typical O-grid structure under valve-bottom

Blocking adjustment

The resulting blocking structure for the case geometry lacks of some adjustment need to prevent from mesh compenetration and bad quality:

- changing of nodes distribution on O-grid edges to create the right boundary layer on liner, intakeDuct and valve;
- moving of vertex of inner blocks to obtain the right block orientation according to geometry;
- splitting of edges according to boundary geometry in order to prevent from cells compenetration;

The first type of adjustment is provided by The Edge Params option that allows you to modify the mesh parameters in a detailed manner by specifying various bunching laws and the node spacing along any particular edge. Each edge has several parameters that determine the spacing of the mesh along the edge. Here it has chosen simply to act on number of nodes, the meshing law, initial length at the beginning/end of the edge (ratio) as in figure 4.33.

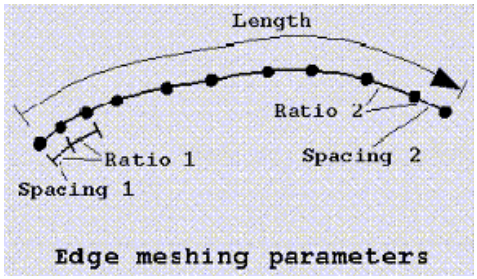


Figure 4.33. Parameters that determine the spacing of the mesh along the edge

A good way of proceeding is choosing an arbitrary number of nodes, not too many, for each group of parallel edges in order to understand the next adjustments to do.

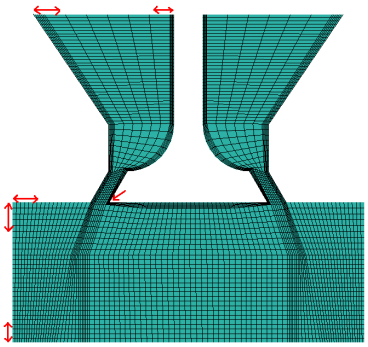


Figure 4.34. Mesh section after a random choice of nodes per edge

The result is shown in Fig. 4.34. A non-smoothed transition between different block regions and lacks of smooth layering within the boundary layer blocks. Red arrows specify where the adjustments have to be performed. The problem is resolved easily on inlet (Fig. 4.35), piston (Fig. 4.36) and cylinderHead (Fig. 4.37):

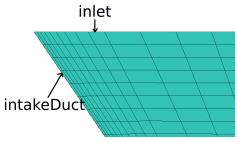


Figure 4.35. intakeDuct layer

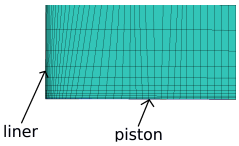


Figure 4.36. piston layer

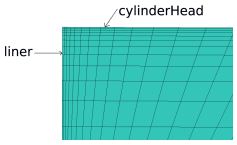


Figure 4.37. cylinderHead layer

The most critical part to adjust is the near valve region (Fig. 4.38).

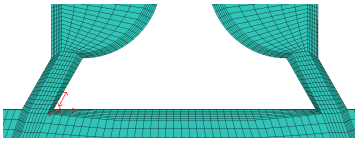


Figure 4.38. Figure shows the most critical region to adjust. Red arrows identify on which edges the spacing will be modified.

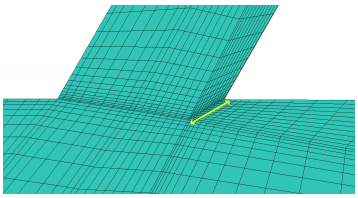
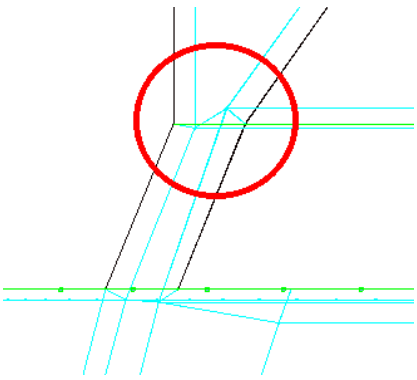


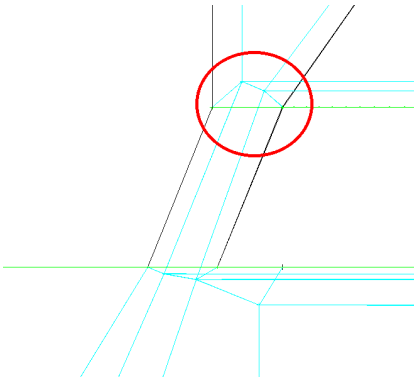
Figure 4.39. Layer around valve after adjustment

In order to realize a quite good layer transition from valve-side to valve-bottom I decide to diminish the initial spacing near valve side (from the bottom to the top) and the spacing of valve-bottom (in radial direction) as well. Moreover an increase of nodes on the edge specified by the yellow arrow in Fig. 4.39. The results, shown in Fig. 4.43 does not seems really good as the user expected: smoothness does not get better, small layer thickness into inner volume and not near wall. The user does not worry about that because another important step has to be done: mesh smoothing is strongly recommended to obtain a good mesh as in Fig. 4.44.

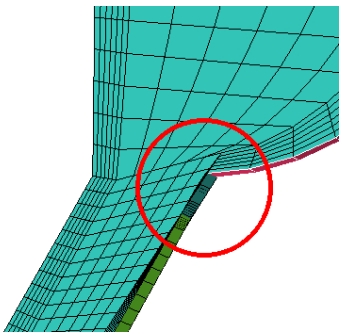
The second type of adjustment is provided by The moving vertex option that allows you to modify the position of the vertex belonging to a block. It's possible to move one or more vertexes simultaneously along a specified direction or basing on the position of a reference vertex. This step allows to prevent from mesh compenetration. In Fig. 4.40 is shown how much the position of a vertex affect the automatic generation of hexaedral mesh in ICEM.



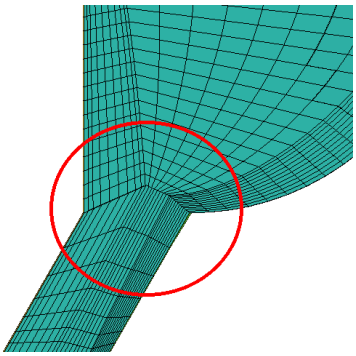
(a) *wrong position of vertex.*



(b) *right position of vertex.*



(c) *mesh due to wrong position of vertex.*



(d) *mesh due to right position of vertex.*

Figure 4.40. Comparison between valve block with two different position of vertexes near valve-top patch. With red circle is specified the zone of analysis

The latter type of adjustment is provided by the Splitting edge tool, in particular by automatic linear option that allows you to project edges, onto the associated geometry, according to the node spacing previously given. It's important underlying that ICEM default options give to edges only 2 nodes (no discretization on edges) and thus if user try to split edge nothing will change.

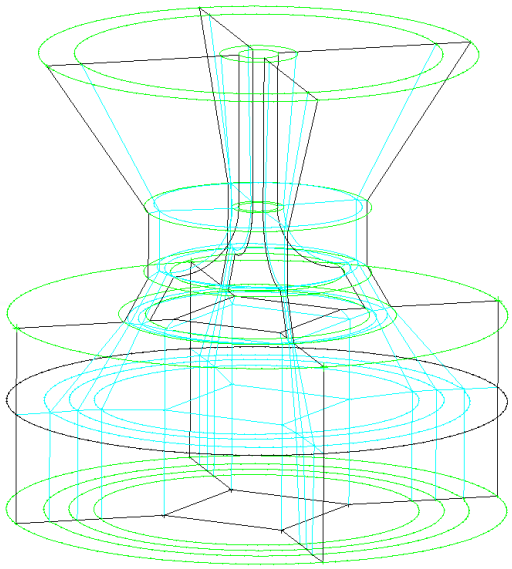


Figure 4.41. Edges splitted

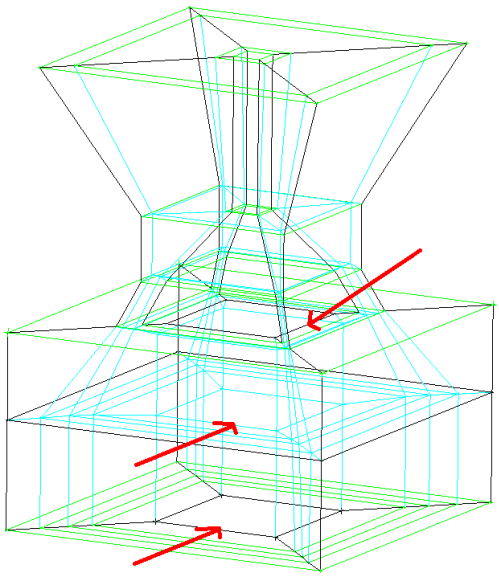


Figure 4.42. Edges not splitted

In many cases this adjustment prevent from the mesh overlapping and allows to orient mesh perfectly with geometry. Both figures 4.41, 4.42 shown that the uniques edges not modified are those of the inner O-grid block, identified by the red arrows.

Pre-mesh definition

ICEM allows to view the meshing result, without converting it in structured or unstructured, through the pre-mesh tool. It generates a temporary mesh file, thus provides a useful way to verify the quality of mesh in few seconds since after only the first definition of block it is hardly difficult reach a good quality, especially in complex geometry. All the previous consideration made upon block have been done thanks to pre-mesh. Thus pre-mesh allows many adjustment on blocking, pre-smooth and the creation of both the unstructured mesh file (.uns) and multiblok structure file (.multiblock). In this case, since the simulation will be done on OpenFOAM the choice fall into the first one, as explained in Sec. 4.3.

Mesh definition: check and adjustment

After the creation of unstructured mesh three more steps must be done:

1. A **check mesh**. Even if this check is provided also by OpenFOAM tool `checkMesh` in higher strictly way, it's worth make it in ICEM before translating the .uns file in .msh file (fluent mesh file) to import in OpenFOAM. The Check Mesh option allows you to locate problems within the mesh that will usually lead to failure when translating or running the solution. The most common problems in this case are:

   - **Volume orientations**, finds elements where the order of the nodes does not define a right-handed element; it can be fixed automatically by ICEM.
   - **Penetrating elements**, checks for surface elements that intersect or pass through other surface elements;
   - **Overlapping elements**, refers to surface elements that occupy part of the same surface area, but don't have the same nodes. This could be surface mesh that folds on to itself. This will also find elements that are at an angle of up to 5 degrees from overlapping each other;

   Penetrating and overlapping elements occur especially when block adjustment have not been performed as explained in sez. 4.3.1 or association between geometry and block is missing and/or is wrong. Thus checking these steps again is useful if a mesh error occurs.

2. A **Smooth Hexahedral Mesh Orthogonal** application. The unstructured hexahedral smoother relaxes unstructured hexahedral meshes in order to obtain smooth grid lines orthogonal to the boundary as well as smooth grid angles and transitions in the inner volume. It first smooths the surface mesh recognizing the topological boundary edges. If the number of volume smoothing steps is greater than 0, after each surface smoothing step the inner volume will be adjusted by performing 1 volume smoothing step. After the surface smoothing has been finished, the inner volume will be smoothed (according to the number of volume steps set). The mathematical basis is that of an elliptical differential equation of the form:

   $$\nabla^2 \mu = f \tag{4.4}$$

   where f is the "control function". It can be proved that by using the elliptical operator ∇^2 , smoothness of the mesh will be achieved. The control function f will be specified so that the smoothed mesh will obtain certain characteristics, such as orthogonality and layer height of the first layer.

3. A **Smooth Mesh Globally** application. This option allows you to automatically improve the quality of the mesh elements. Different smoothing algorithms are

available depending on which mesh type is loaded. Mesh can be smoothed with respect to a particular quality criterion and with a specified number of iterations to achieve a given quality level.

The two last option must be applied accordingly to the previous enumeration. Since Hexaedral smoother help in reaching a smooth mesh without taking into account the mesh quality, the user will have to use later the smooth mesh globally option to reach a good quality up to a chosen value in a specific quality metric.

In order to obtain the mesh shown in Fig. 4.44 from mesh of Fig. 4.43 the set-up used for Hexaedral smoother is:

- Number of iterations for the solver: 10 iterations on surface and 5 iteration on volume;
- Orthogonality smoother on surface, laplacian smoother on volume, thus solve a pure diffusion equation $\nabla^2\mu = 0$ on all nodes of mesh;
- Freeze option onto all patches to freeze all boundary node locations;
- smooth all curves in smooth along curve option;
- Treat unstruct nodes with lapalacian smoother;
- Surface fitting in order to constrains boundary nodes to the true geometry surfaces;

After the Hexaedral smoothing, the smooth mesh globally option is applied on volume and surface up to a 0.3/0.4 value of skewness and quality metrics, normalised to unit, in ICEM.

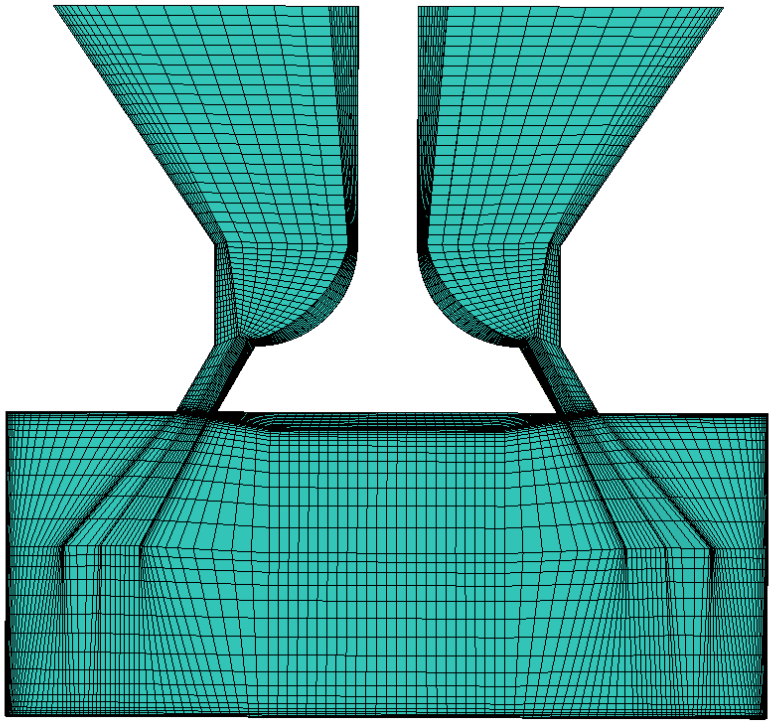


Figure 4.43. Figure shows mesh after spacing adjustment on all edges aforementioned

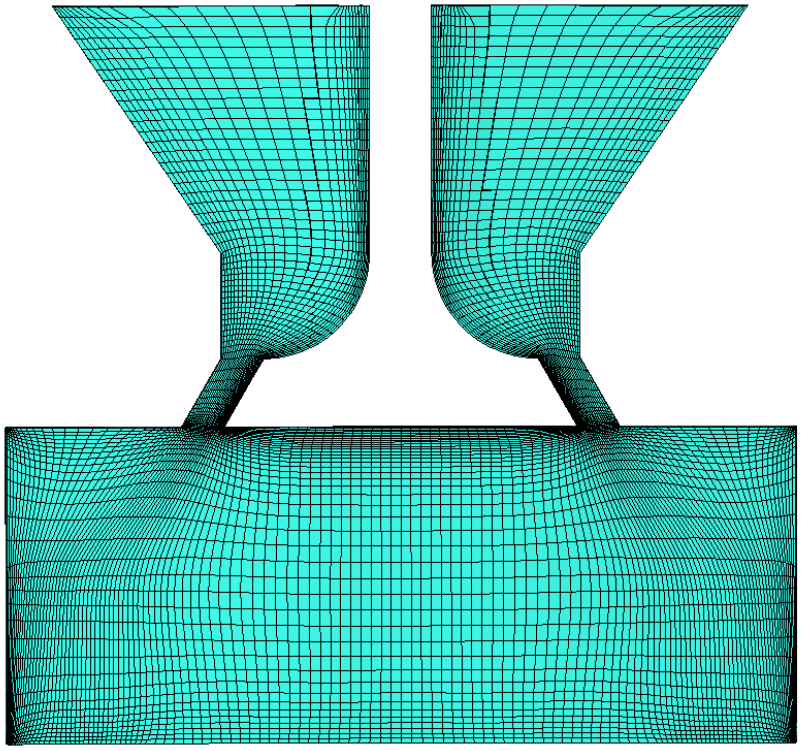


Figure 4.44. Figure shows mesh after laplacian smoothing on volume

4.3.2 Transparent Combustion Chamber case

The transparent combustion chamber (TCC) shown in Fig. 4.45,

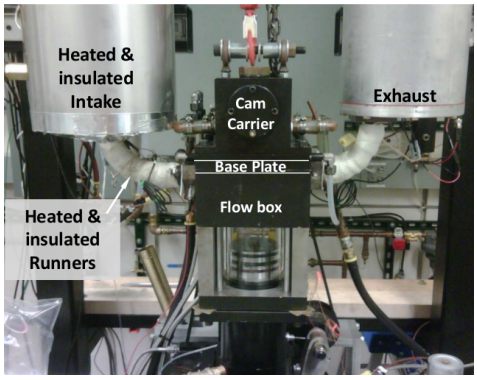


Figure 4.45. TCC-III equipment at University of Michigan

from Engine Combustion Network ECN [20], has been chosen as second test-case. TCC was developed for the specific purpose of supporting the development and validation of LES approaches.
The engine is made up of a two-valve head with simple intake and exhaust port/runner geometries and a pancake-shape combustion chamber.

The TCC tools was developed in 1990-1995 with TCC-0 geometry by General Motors Research department and studied in [21, 22, 23, 24, 17]. The data were taken with the purpose of Particle Image Velocimetry (PIV) development, empirical observations of Closed Cranckcase Ventilation (CCV), and RANS development.

Then research on TCC were carried on by University of Michigan in period 2010-2012 with TCC-I (Kuo, Tang-Wei et al. [25], Sick [26]) geometry (for simulation only) and TCC-II geometry (Abraham et al. [27]). Compared to the TCC-0, the TCC-II has a completely new intake and exhaust systems, different crank-case, and different dynamometer. Testing revealed the intake valve guide was worn, such that later oscillations occurred during lift. It was also discovered that the valve seat had two rather than one angle, and thus had different discharge coefficients. Early tests during this period would sometimes experience sudden and random engine speed variations during testing, due to faults in the dyno powerpack as documented in the publications. These "faults" were used as opportunities to search for their impact on the flow.

Afterward TCC study went on in University of Michigan during the period 2013-2014 with TCC-III geometry. The TCC-III engine was refurbished with new hydraulic valve lifters, new valves (identical design), and new four-angle valve seats. The in-

take and exhaust systems are prepared for firing tests, including gaseous-fuel and a nitrogen-dilution critical-orifice metering systems, flame arrestors have been added at the plenum inlet/outlets, and water-cooled exhaust-pressure transducers. Thus, new engine geometry files are required both for the LES and 1-D (GT Power) simulations of the TCC-III engine (Montorfano, Piscaglia, and Onorati [12]). Compared to the TCC-II, improved operating procedures were instituted to assure engine operation repeatability. The data set contains six operating conditions, four measurement planes, repeated tests, and more cycles per test. Thus the most important feature changed from period 1990-2014 was the valve-seat. In figure 4.46 is shown the valve-seat anthology of valve-seat from 1990 to nowadays.

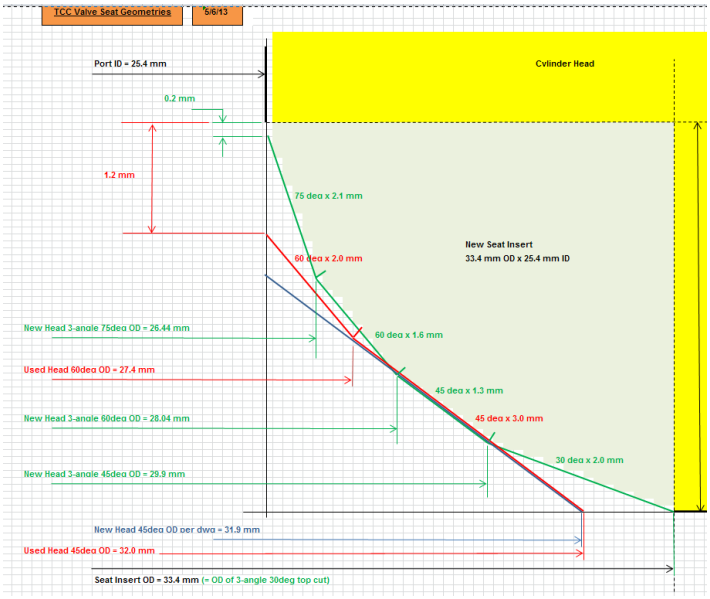


Figure 4.46. Valve-seat anthology from 1990 to 2014. Green lines represent the TCC-III valve-seat

My case study comes down on TCC-III, thus the configuration with 4 angle valve-seat, which whole geometry is shown in figure 4.47. In TCC case the parts involved in topological changes of moving geometries are:

- Piston, as in flat top cylinder;
- liner, whose length increase or decrease according to piston motion;
- non-conformal interfaces with arbitrary mesh interface (AMI) implemented in OpenFOAM to enable simulation across disconnected, adjacent, mesh domains. Under intake/exhaust valve geometry;

- intake and exhaust valve, whose lift has been defined in a table function of crank angle. Then these tables are invoked in th file `engineGeometry`, the same for piston motion, in order to provide valve motion.
- spark plug and detachFaces. On The first a slidingInterface is applied one time through the use of `stitchAndSplitMesh` in order to connect fluxes between cylinder and spark plug region, that after its application becomes only one region. Thus slidingInterface is applied on a non-conformal static interface. In the second case between the two detached region that arise when attachDetach topology modifier is triggerd at valve closure/opening event, hence applied on a conformal interface.
- Baffles, created during the meshing, whose presence need to provide valve motion and closure. The baffles are converted in a STL file and grouped in `trisurface` directory. Afterward baffles are invoked by `topoSetDict` to create `faceSet` and `cellSet` for those mesh region in which a topological change is requested.

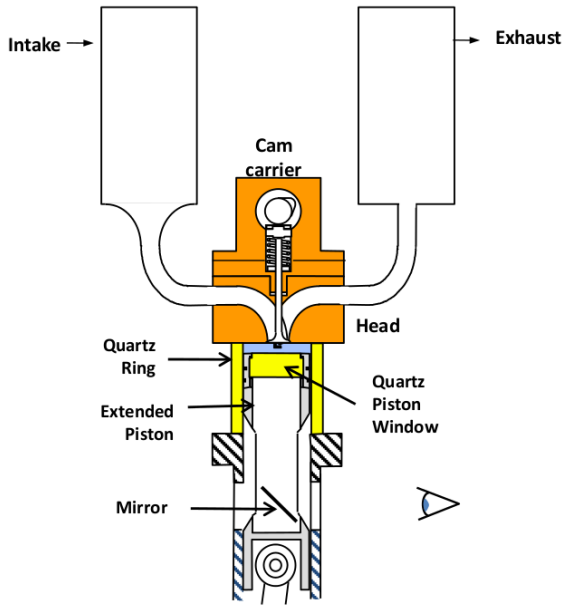


Figure 4.47. whole TCC-III equipment's geometry

Geometry definition

As in flat-top cylinder case the meshing begin from a IGES geometry file. The geometry taken from ECN describe the whole equipment (plenum and TCC), however the

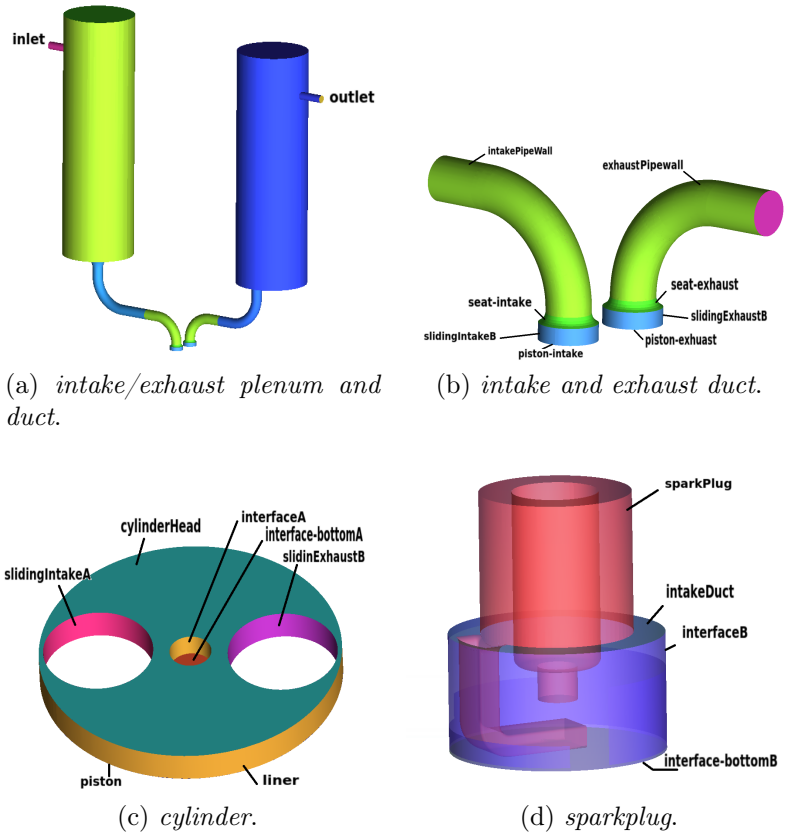
meshing procedure has been developed splitting geometry in three parts:

1. intake-parts and exhaust-part, figure 4.48a; in particular, as starting point block structure is realized on intakeDuct and intValve and then will be created on exhaustDuct and exhValve using mirroring of blocks. Afterwards blocks of intake and exhaust plenum are merged to them.

2. cylinder (Fig. 4.48c);

3. sparkplug, (Fig. 4.48d);

Only blocks of plenum and ducts has been merged directly in ICEM because between them no-one non-conformal interface have been used, thus only two fluid region, one for intake and one for exhaust, are generated.



(a) *intake/exhaust plenum and duct.*

(b) *intake and exhaust duct.*

(c) *cylinder.*

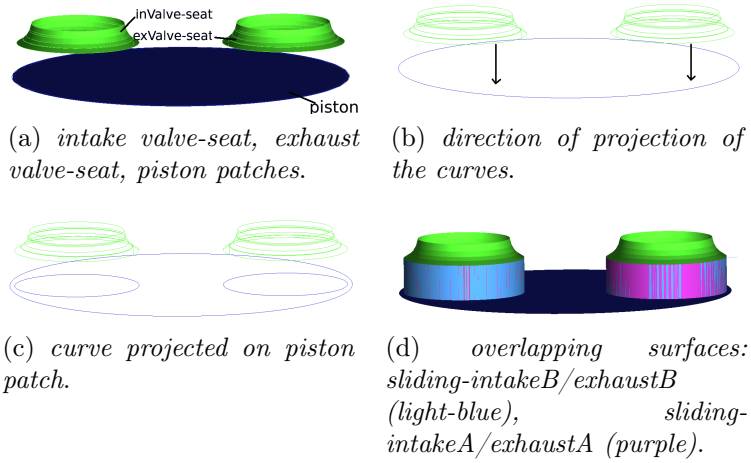
(d) *sparkplug.*

Figure 4.48. Figures show in which parts the whole geometry has been splitted to face the meshing procedure

Many adjustment has been done on starting geometry downloaded from [20]:

- four new surfaces, slidingIntakeA, slidingExhaustA , slidingIntakeB, slidingExhaustB (between ducts in Fig. 4.48a and cylinder in Fig. 4.48c), has been created in order to have non-conformal interface for AMI, to provide valve motion.
- four new non-conformal surfaces between cylinder (Fig. 4.48c) and spark plug (Fig. 4.48d) in order to have a good layering under ground electrode. This way ensures the right application of `layerAdditonalremoval` modifier on piston patch, and guarantees a different spacing and strategy of blocking between cylinder and spark as well. Afterwards when the two mesh are completed they will be stitched obtaining a non-conformal static grid. The reason why a slidingInterface is preferred is that 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.
- 2 baffles, for each valve, has been created to provide the forward extraction of stl file by `topoSetDict`, for `layerAdditionalRemoval` and `attachDetach` topology modifier.

The patches on which AMI rely on are realized in ICEM starting from the diameter of valve-seat visible in Fig.4.49a. The curve representing the intersection between cylinderHead and valve-seat has been projected on piston surface through "the project curve on surface" tools under **geometry-create/modify curve** (Fig. 4.49c). Then 2 surfaces for intake, slidingIntakeA (side cylinder) and slidingIntakeB (side intake) has been created extruding the curves, from piston patch to valve-seat in z direction, through "curve driven" tools under **geometry-create/modify surface**. It is necessary that 2 surface overlaps geometrically but they must own a different label name (Fig. 4.49).



(a) *intake valve-seat, exhaust valve-seat, piston patches.*

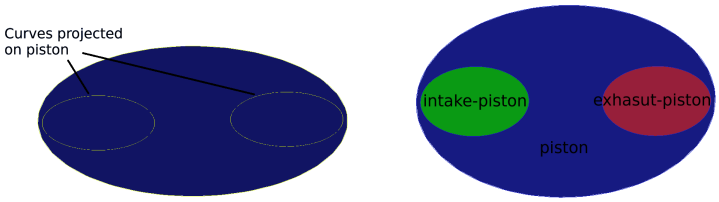
(b) *direction of projection of the curves.*

(c) *curve projected on piston patch.*

(d) *overlapping surfaces: sliding-intakeB/exhaustB (light-blue), sliding-intakeA/exhaustA (purple).*

Figure 4.49. Figure shows the steps to create the 4 surfaces

Afterwards it is useful to split piston patch in 3 parts: piston which belong to cylinder part, piston-intake and piston-exhaust belonging to intake and exhaust respectively. To do that has been used the "segment/trim surface by curve" tools under **geometry-create/modify surface** using the curve projected on piston patch. Then under **tree-parts/create part/create part by selection**, selecting each new trim surface intake-piston and exhaust-piston has been created, as shown in Fig. 4.50b



(a) *curves use to trim the piston patch.*

(b) *splitting of piston patch in 3 sub-parts.*

Figure 4.50. Splitting of piston patch in intake-piston, exhaust-piston and piston

The non-conformal interfaces between cylinder and sparkPlug rely on the following geometry surface: interfaceA, interfaceBottomA (side cylinder) and interfaceB, interfaceBottomB (side sparkPlug). After the application of `stitchAndSplitMesh` these surface will disappear. The same extruding procedure, from curve, has been applied.

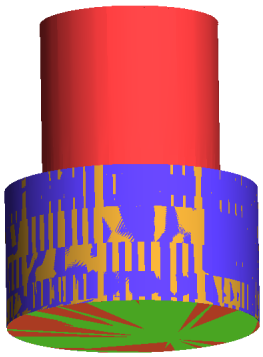


Figure 4.51. Figure shows in red the sparkPlug patch and the overlapping surfaces: interfaceBottomA (red), interfaceBottomB (green), interfaceA (orange), interfaceB (blue)

Finally baffles, from which export stl file, have been generated for both valve: exhvalve-topFaces, invalve-topFaces, exhaustDetachFaces, intakeDetachFaces. exhvalve-topFaces and invalve-topFaces has been created from the revolution of one curve around the valve axis. The curve uses for this purpose is not made of only one segment (Fig. 4.53) from sliding-intake/exhaust but by more segment as shown in Fig. 4.52 Comparing Fig. 4.54 and Fig. 4.55 it is possible understand the reason why the choice of one curve made of a unique segment (4.53) is not allowed. Thus the curve, in order to not cross the valve seat, has to follow his geometry especially for the formers two angles. After those can be made a single segment to the bottom of valve-stem . This particular choice will be explained completely in blocking definition due to the further relation between the baffle and the block around valve it was chosen to do.

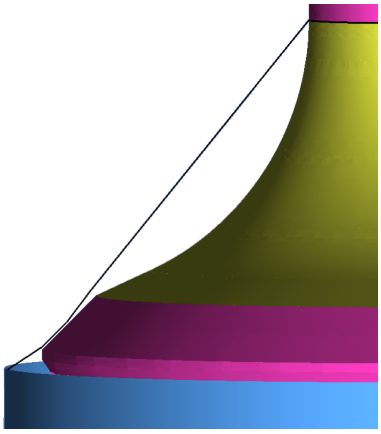


Figure 4.52. curve use to obtain the surface of revolution

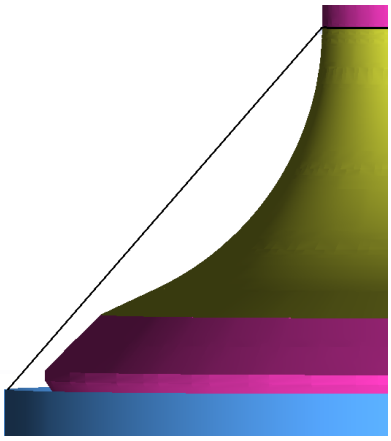


Figure 4.53. wrong curve not use for revolution

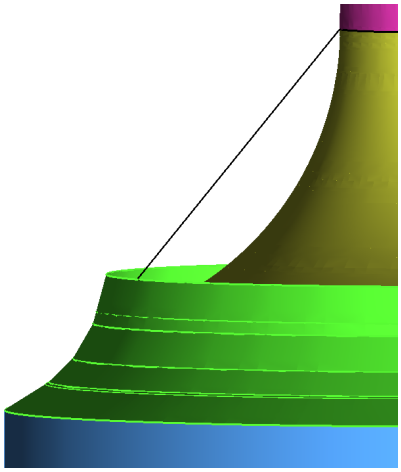


Figure 4.54. curve does not cross seat

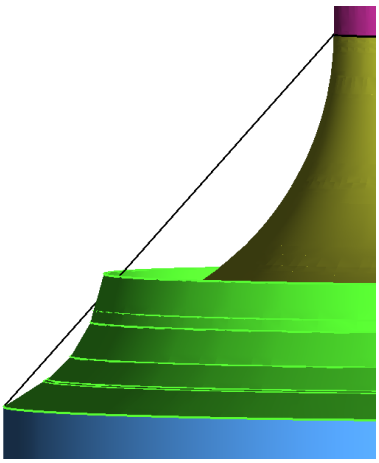


Figure 4.55. curve crosses seat

Then applying the "surface of revolution" tools under **geometry-create/modify surface**, using the aformentioned curve, exhValveTopFaces and intValveTopFaces has been done, as shown in Fig. 4.56.

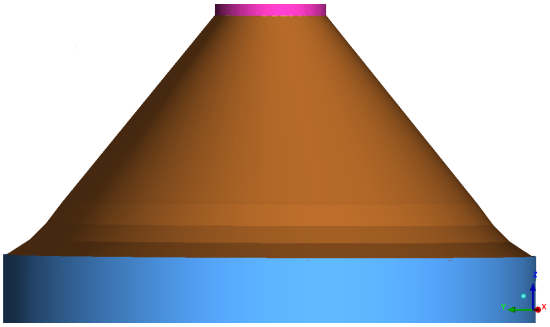


Figure 4.56. TopFaces

It is important underlying that curves does not start from the top of sliding-intake/exhaust patch but a little bit shift downward, as shown in Fig. 4.57

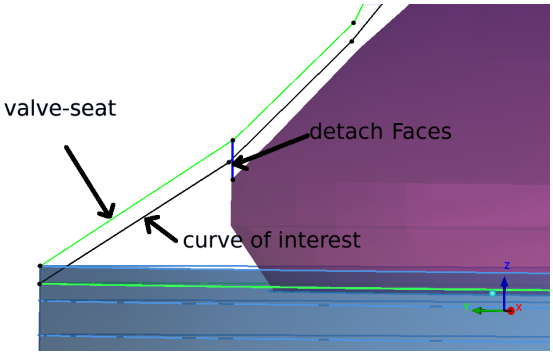


Figure 4.57. near view of curve

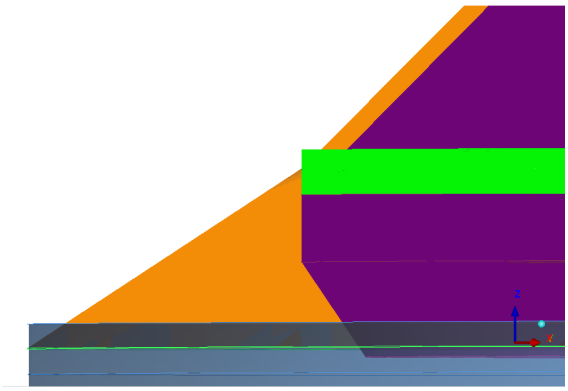


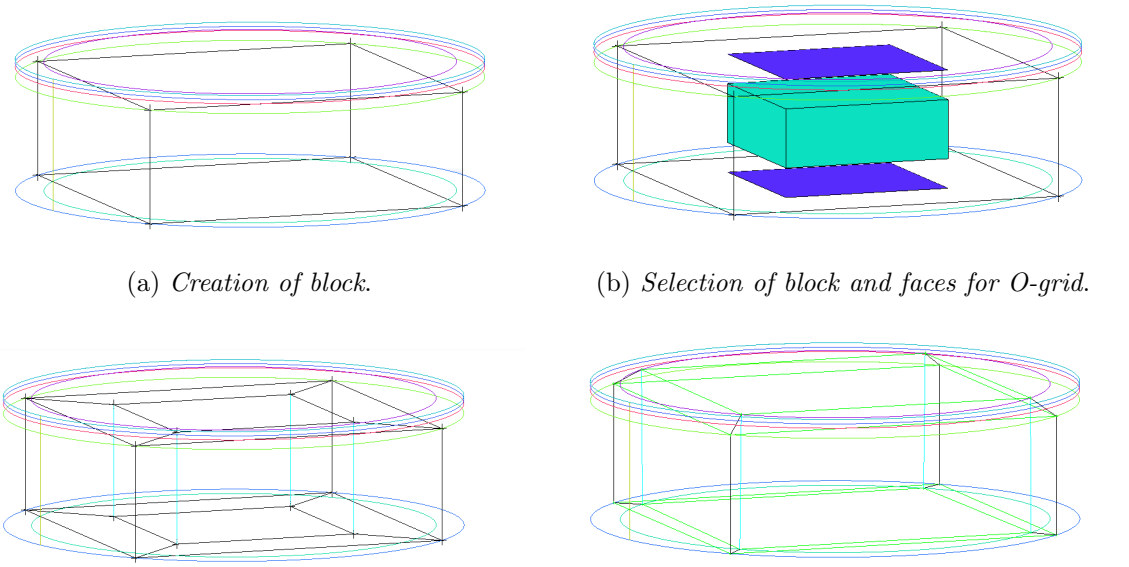
Figure 4.58. Fig. shows the attachDetach in green

In Fig. 4.57 is shown also the curve use to create exhaust/intake-detach. The surface will be a surface of revolution around the axis of the valve. Thus the final surface will cross the exhValve/inValveTopFaces as shown in Fig. 4.58.

Blocking definition and adjustment for intake and exhaust geometry

The blocking procedure for intake and exhaust follows these steps:

1. Construction of block from intake-piston to vale-side/valve-top intersection. Beginning from a single block (Fig.4.59a) between valve-bottom and intake-piston, an O-grid has to be created (Fig.4.59c). Then edges has been associated to the corresponding curves as shown in Fig. (Fig.4.59d).



(a) *Creation of block.*

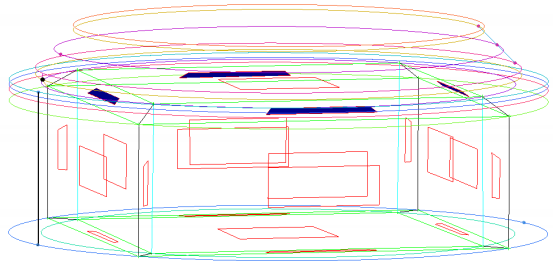
(b) *Selection of block and faces for O-grid.*

(c) *O-grid block under valve.*

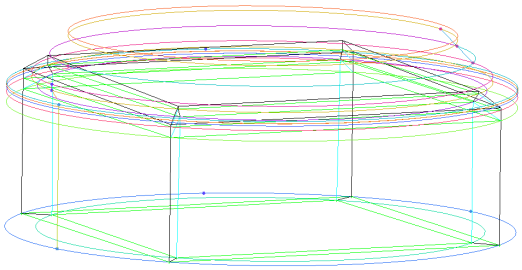
(d) *Association of edges to corresponding curves.*

Figure 4.59. Fig. shows the steps to obtain O-grid under the valve

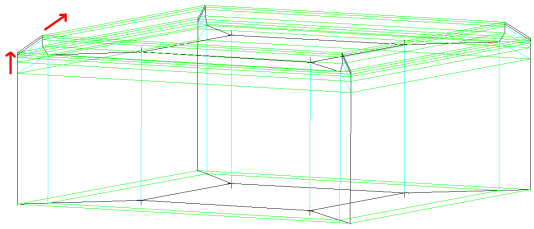
The idea is create an O-grid block whose top-face is oriented with the valve-seat geometry (Fig.4.59d), in order to obtain an oriented mesh in jet zone. Afterwards the most external faces of the top of block has been extruded in z-direction as shown in Fig. 4.60d. Then, the block faces (representing the attachDetach baffles) has been extruded along the valve-side until the valve-side/valve-top intersection (Fig. 4.60f). The extrusion tools (under **block/create block**) can be used or in interactive mode (after selection of faces the user drag the faces until the reaching of wanted height) or in extrude along curve mode (after selection of faces the user have to chose a reference curve for extrusion direction and an end point for extrusion).



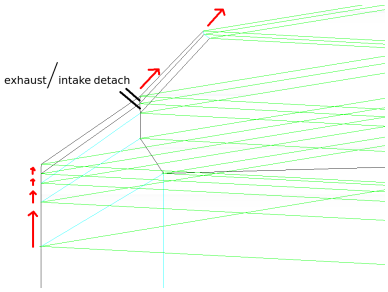
(a) *Selection of faces to be extruded.*



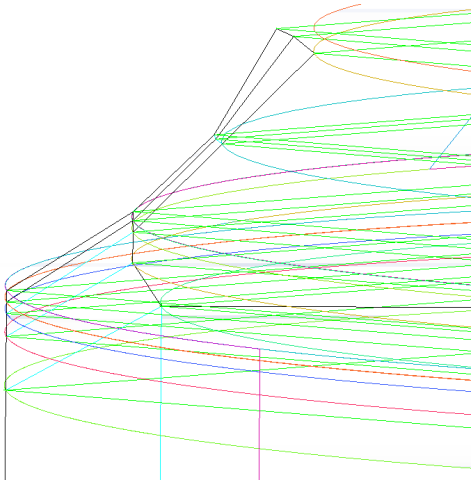
(b) *First extrusion done.*



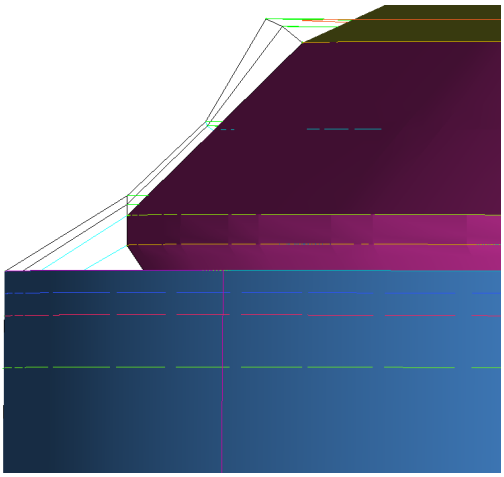
(c) *Direction of extrusion with red arrow.*



(d) *Direction of extrusion along valve side and the edges correspondong to the block faces to be associated with exhaust7intake detach baffle.*



(e) *Blocks after firs step blocking procedure.*



(f) *Relative position between block and valve-side patch.*

Figure 4.60. Fig. shows the steps to obtain the block from the piston-intake to valve-side/valve-top intersection

2. Construction of block from valve-side/valve-top intersection to valve-top/valve-stem intersection, under inValveTopFaces; this is the most critical part of blocking procedure because boundary layer on intakeDuct and valve, either a good cellSet for valve motion rely on how the block between inValveTopFaces and valve-top have been done. Since both intakeDuct and valve need a boundary layer, whose dimension are the same of block between valve-side and valve-seat, the blocks shown in Fig. 4.60e near valve-side has to be extended along the two patches aforementioned using a "key brick" block as show in Fig. 4.61

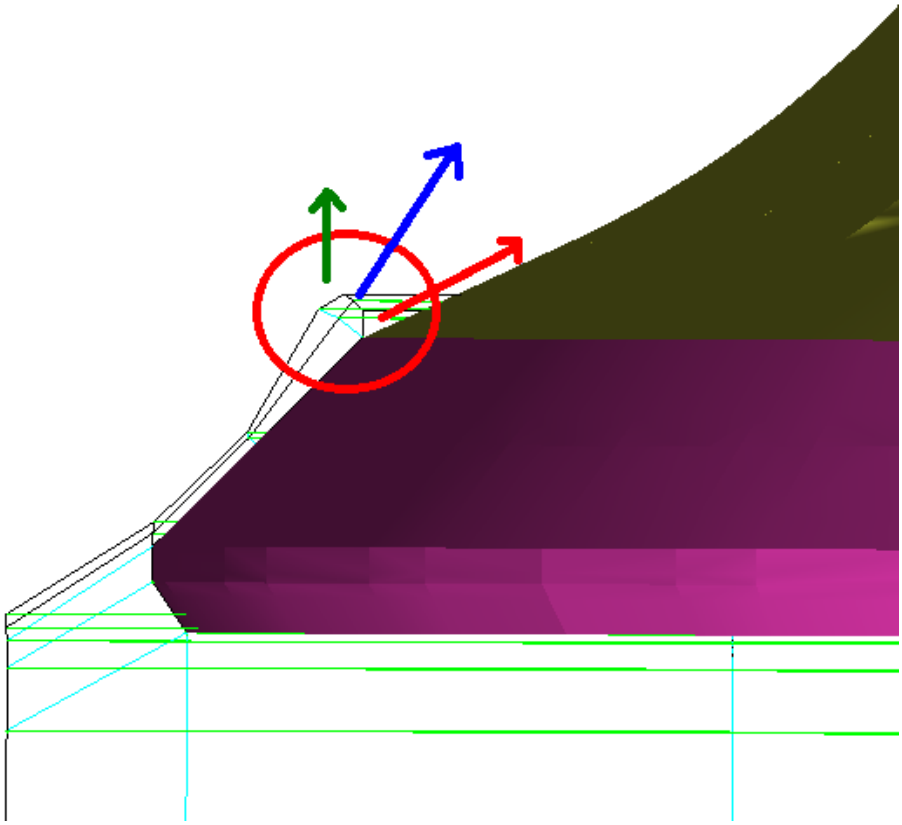


Figure 4.61. Fig. shows the faces involved in the extrusion

It is important to underlying tha the procedure has been done on only one side (over 4 existing side). Thus this step must be repeated for each side and vertexes belonging to each side must be merged to the other at the same height. The former part of extrusion involves faces whose extrusion direction is represented by red and blue arrow. Faces on the tail of blue arrow are initially extruded along z-direction (Fig.4.114a)

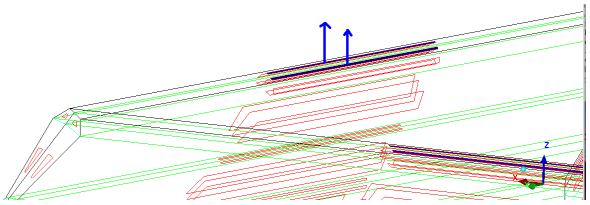


Figure 4.62. Fig. shows the faces must be extruded and the extrusion direction

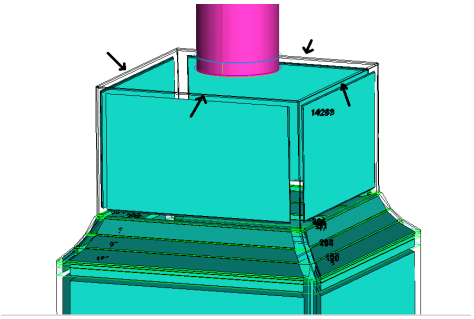


Figure 4.63. Fig. shows block created with the extrusion

The black arrow in Fig. 4.63 shows the edged has been associated to curve near valve-stem/valve-top intersection. The result is shown in Fig. 4.64

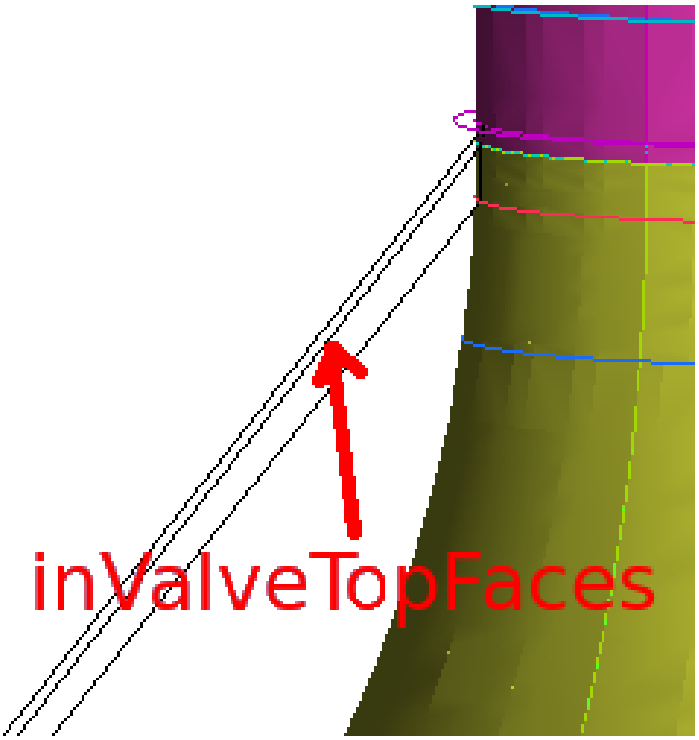
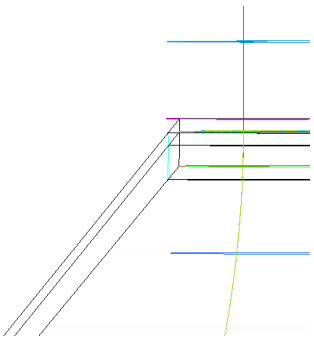
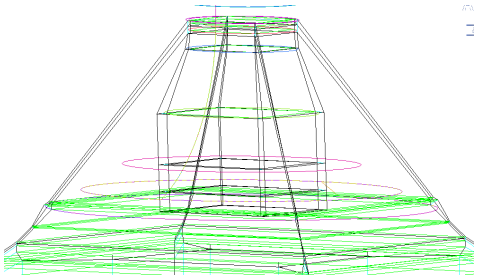


Figure 4.64. Fig. shows the association of edged to curves belonging to valve-stem and valve-top

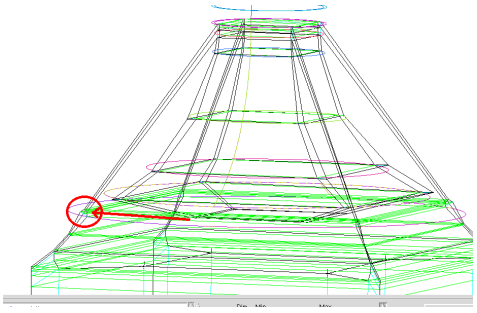
Since now that there are the faces corresponding to inValveTopFaces it is possible to associate to the geometrical surface, as seen in Sec. 4.3.1. Afterward blocks along valve-side ha evbeen created, splitting block near valve-top/valve-stem (Fig.4.65a) intersection and extruding the bottom face of new block in z direction. Then extruded blocks hebeen splitted in 5 sub-blocks, to fit the best as they can the valve top geometry, and then each edges' block has been associated to valve-top curve as shown in (Fig.4.65c). Finally the vertexes pointed out with the tail of red arrow in Fig. 4.65c has been merged with the corresponding vertexes (valve-side/valve-top intersection), within the red circle.



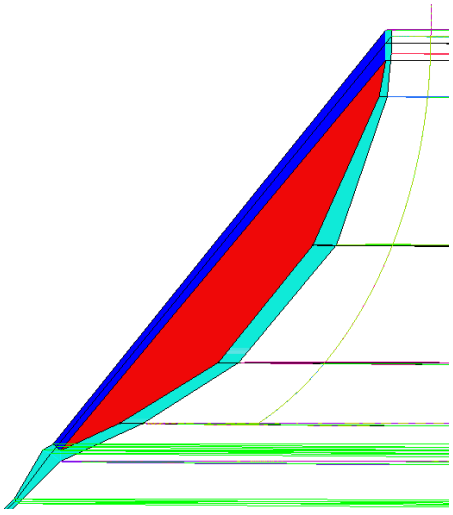
(a) *Split of block near valve-stem/valve-side intersection.*



(b) *Extrusion of block.*



(c) *Association of block to valve-top curves.*



(d) *Fig. shows the block in light-blue, while red represent a "void" to fill with a particular block structure. Blue block represent a block to split in ordr to create quarter O-grid.*

Figure 4.65. Steps to create boundary blocks for valve-top and inValveTopFaces

The blue block in Fig. 4.65 need to be split in a number of blocks egual to those along valve-top. These will guarantee the creation of quarter O-grid.

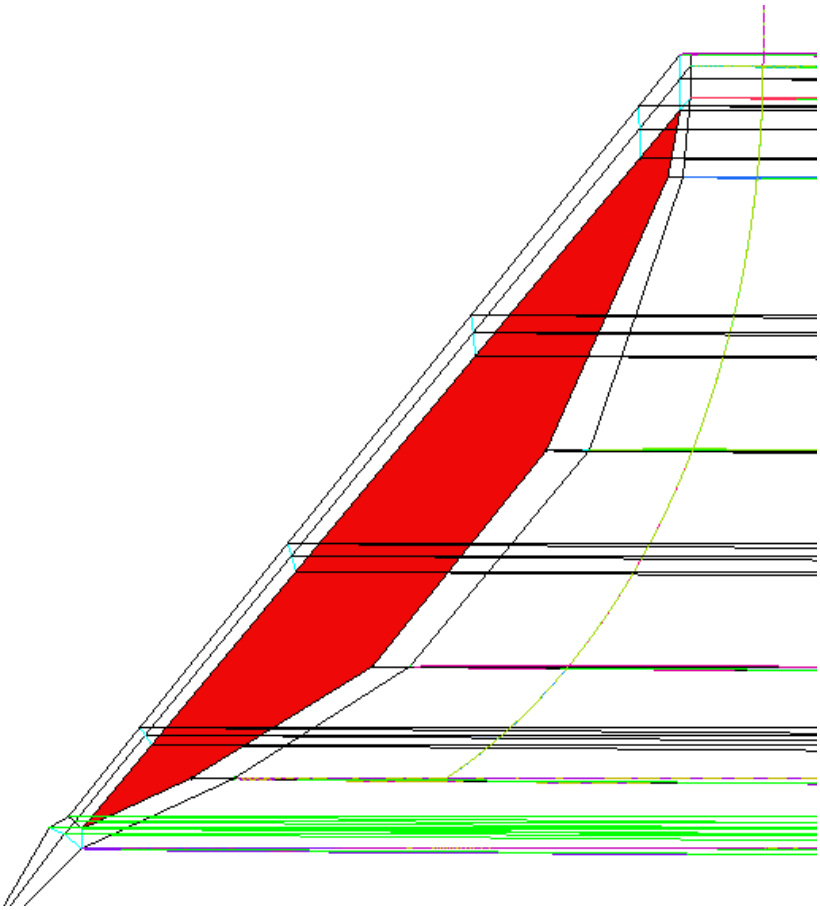


Figure 4.66. Splitting of block along inValveTopFaces

The result is shown in Fig. 4.66. The following step deals with the creation of a particular block structure within the red field shown in Fig.4.65d. ICEM allows to create the advanced topology known as a Y-Block or Quarter O-Grid (under create block-from vertices/faces-Quarter O-grid). This topology is used to fit three Hexa Blocks into a wedge. Select six vertices as shown in Fig. 4.67 to create the quarter O-grid. The three vertices of one side of the wedge must be selected first, in clockwise or counter clockwise order. Then the remaining three vertices can be selected. It is important that the 4th vertex selected should be connected to the 1st, 5th to the 2nd and 6th to the 3rd respectively.

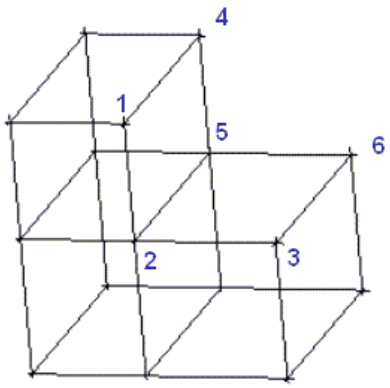


Figure 4.67. Rule to select vertexes for quarter O-grid

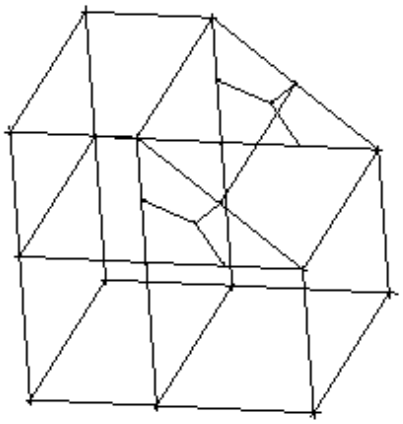


Figure 4.68. Quarter O-grid

This approach can be used similarly between in/exhValveTopFaces and in in/exhValve-top (Fig.4.65d), in order to obtain a block as that of Fig. 4.68. Thus Quarter O-grid has been applied to the top and to the bottom.

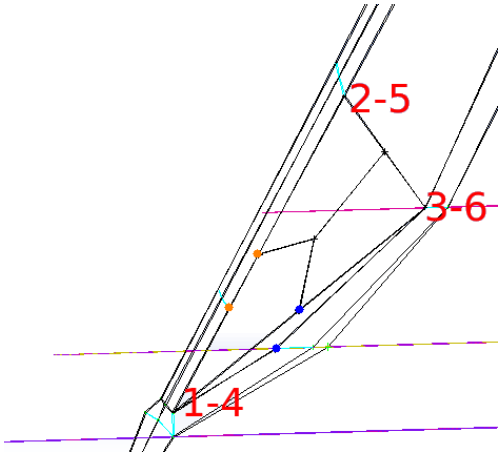


Figure 4.69. Quarter O-grid in the bottom of block

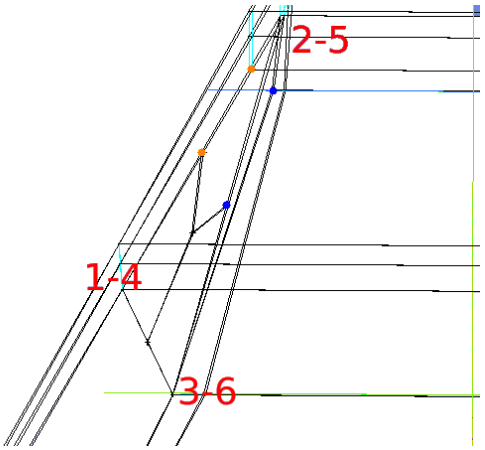


Figure 4.70. Quarter O-grid in the top of block

In Fig. 4.69 and 4.70 are shown the number of couples of vertexes to select and the resulting quarter O-grid block. Vertexes to merge are pointed out with the

same colour. Between bottom and top blocks two more blocks has been created using create block tools from vertices (hexaedral) as shown in Fig 4.71.

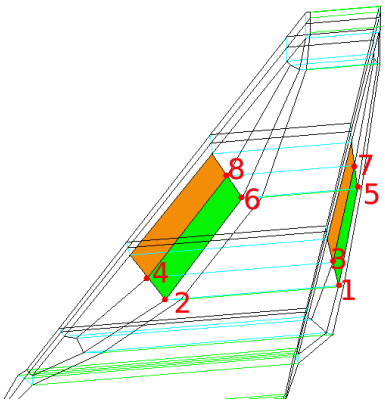


Figure 4.71. Creation of hexaedral blocks from vertices

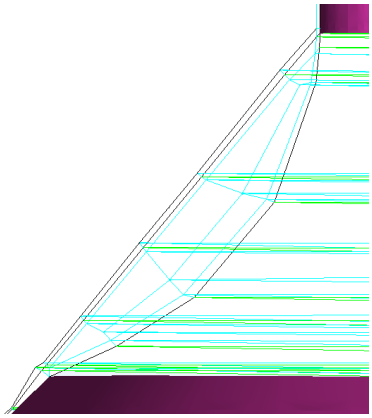


Figure 4.72. Lateral view of final blocks near valve

The blocks are created in the same way both for green faces and orange faces (for 4 side around valve). Then few adjustment has been done on blocks in Fig. 4.71, such as move vertex, to obtain the best block of structure (Fig. 4.72).

3. Construction of block above inValveTopFaces to the middle of valve-stem. Now block must be created over the inValveTopFaces through faces extrusion (Fig.4.73).

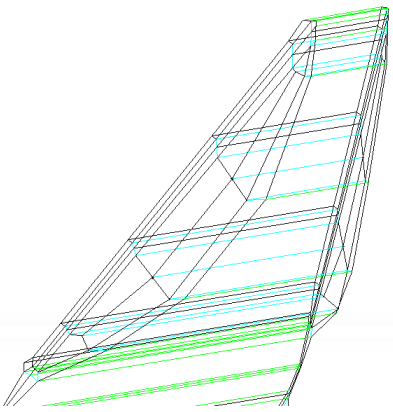
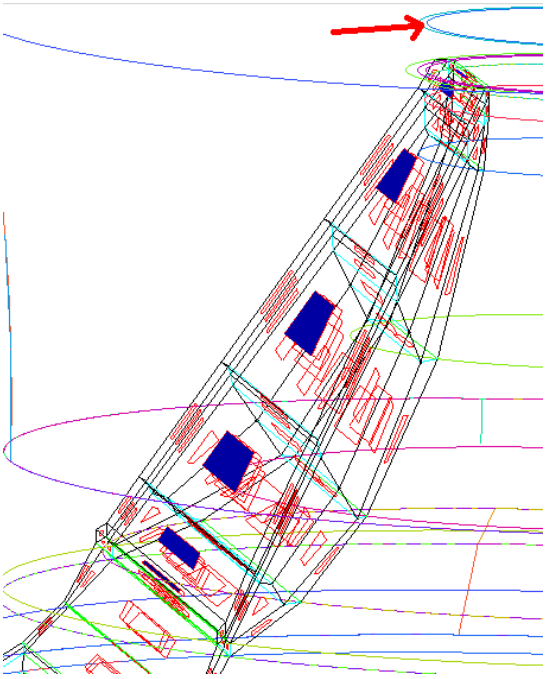
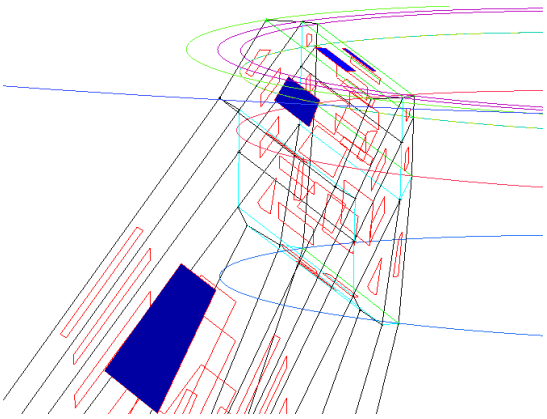


Figure 4.73. Blocks after extrusion of top faces of block in Fig 4.72

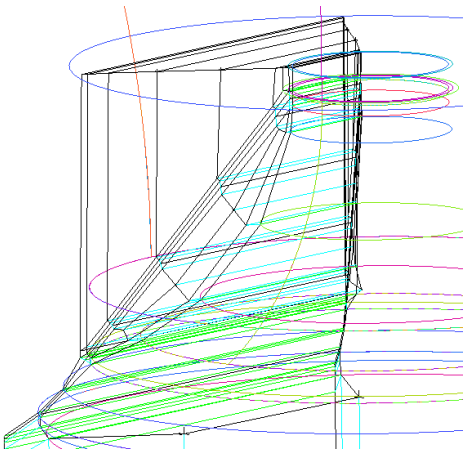
Then the top faces of block of Fig. 4.73 has been extruded in z direction till the curve pointed out by the red arrow inf Fig. 4.74a



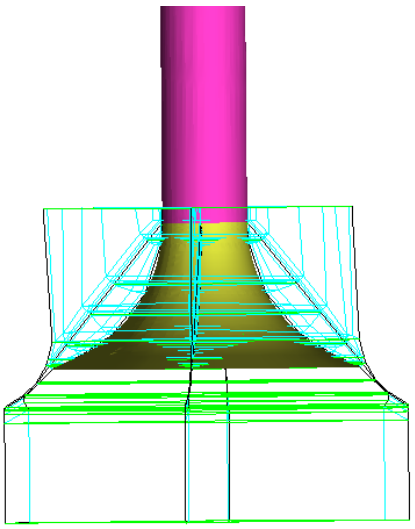
(a) *Fig. shows the faces selected for the extrusion .*



(b) *Near view of face selection.*



(c) *Block after extrusion.*



(d) *View of block around valve-top and valve-stem.*

Figure 4.74. Extrusion of block above inValveTopFaces

4. Construction of elbow of duct; In [28] is shown a strategy for "Hexa Mesh Generation for an Elbow Part". Although it seems to be a good approach, for TCC case is too simple, since are involved much more blocks than in the tutorial, and it does not provide a good quality where the elbow crosses with valve-stem. Thus has been adopt a quite different method. Before proceeding with the blocking on the elbow, two curves are extract from surface of intakePipeWalls with trim surface tools (which generate curves as intersection between the plane use to trim and the trimmed surface). The 2 curves need to associate it has chosen to do between them. These are shown in Fig. 4.75

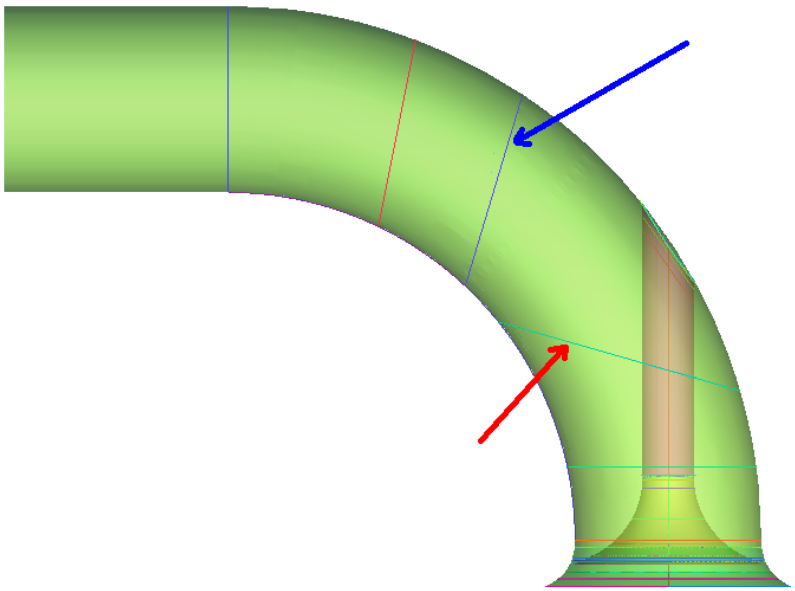


Figure 4.75. Extraction of curves from intakePipeWalls patch

Usually, as in a tutorial, when there is a solid region, for which we are not interested to investigate the physics (e.g conduction), the mesh is created only on his surface. Therefore an O-grid block is used to realize the axial-symmetric mesh around it then the internal block is deleted since it is useless. This was the case of axis-centered valve in sec. 4.3.1. However for TCC case is not worthwile this approach due to poor quality of mesh it would generate near valve-stem-intakePipewalls intersection.

First, the top faces in Fig.4.74 has been extruded to the curve pointed out with red arrow in Fig. 4.75. Afterward the blocking of elbow can start.

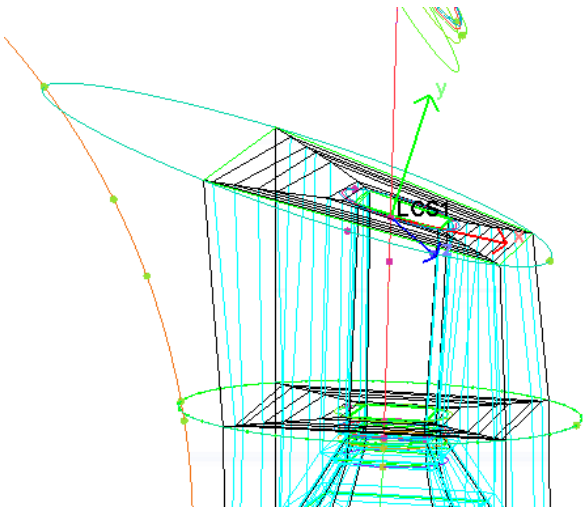
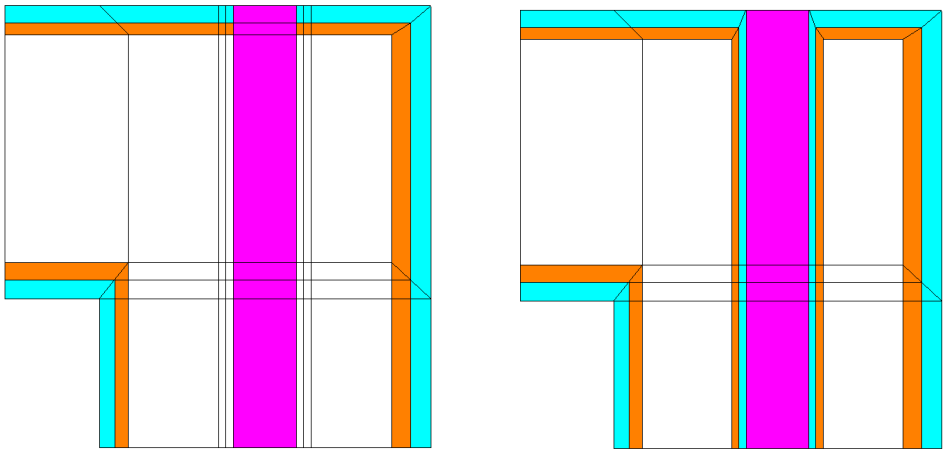


Figure 4.76. Fig. shows the faces extruded to create the block. LCS represent the local coordinate system use to move the points on x-z plane of LCS

The difference between tutorial and TCC is shown in Figs. 4.77a, 4.77b



(a) *Manual block custom made for TCC .* (b) *Block tutorial with automatic O-grid generation.*

Figure 4.77. Synthetic comparison between block section of two approaches. Purple represents valve-stem while light-blue and orange represent the boundary layer block.

As underlying before there is a remarkable difference between the two approaches at valve-stem/intakePipeWalls intersection. In the former case the blocks cross over the valve-stem, at the top, and these allows to have around valve stem blocks whose spacing differs from that of intakePipeWalls, moreover it allows a god quality of mesh. However this approach is not automatic but depends on the experience of user. The latter is automatic in ICEM because relies on O-grid block but it constrains to have the same spacing along intakePipeWalls and valve-stem, and the typical structure of O-grid around valve-stem as well. For that reason it has been preferred chose a non-automatic generation of hexaedral block instead of the most automatic and quick one.

The top faces in Fig. 4.76 has been extruded along the z direction (of global coordinate system). Afterwards the edges has been oriented, as the curve pointed out in Fig. 4.78, using another LCS.

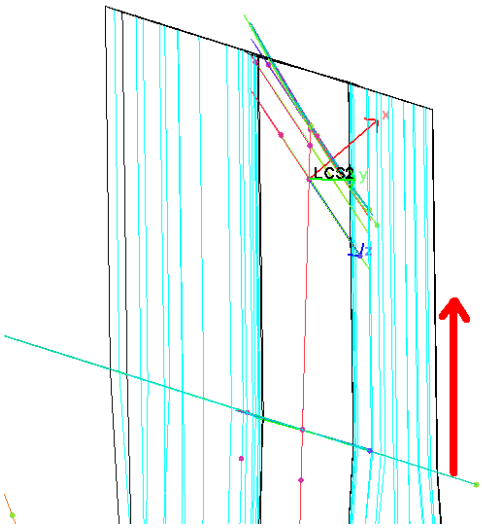


Figure 4.78. Extrusion of block around valve-stem

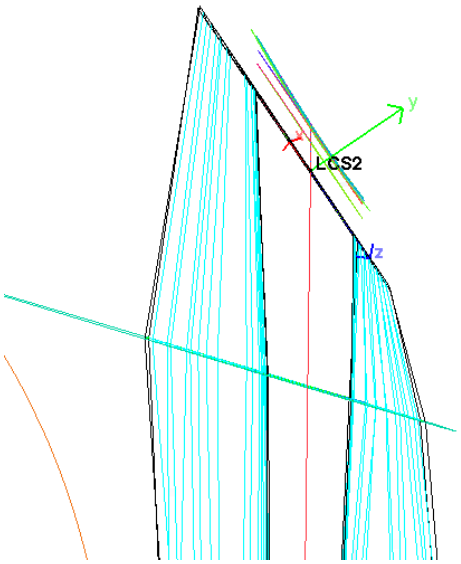
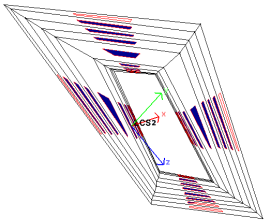
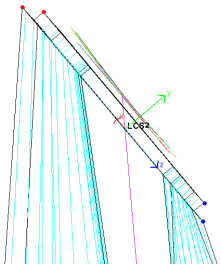


Figure 4.79. Orientation of block as LCS2

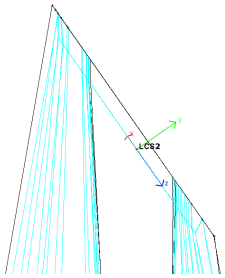
From the block state of Fig. 4.79, two more extrusion has been done in order to obtain the particular block of Fig. 4.77a.



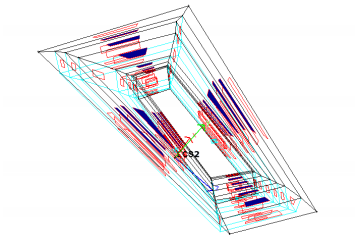
(a) *Selection of faces for first extrusion.*



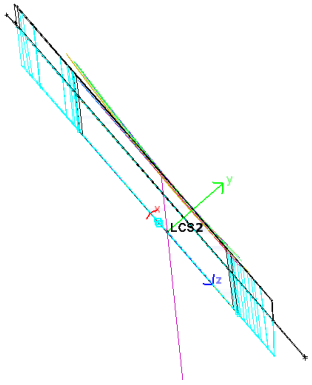
(b) *Faces extruded and coloured point to be merged.*



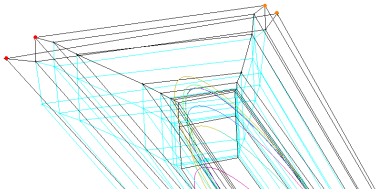
(c) *Lateral view of block after vertexes merging.*



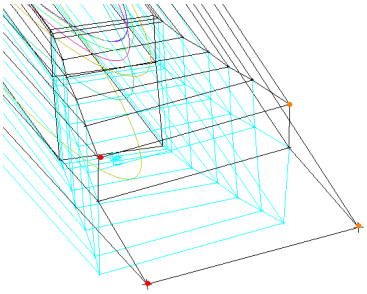
(d) *Selection of faces for second exrusion.*



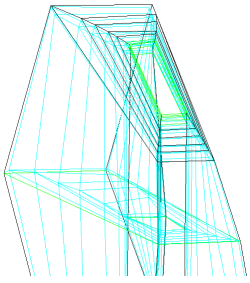
(e) *Faces extruded.*



(f) *Coloured point to merge at the top of the block.*



(g) *Coloured point to merge at the bottom of the block.*



(h) *Final block.*

Figure 4.80. Show the steps to obtain the particular blocking shown in Fig. 4.77a

Extrusion of the left face of top block (Fig. 4.80h), to realize the final part of duct, seems to be a bad solution since does not provide a boundary layer: it does not have the typical structure of O-grid, as pointed out in Fig. 4.81.

Thus this block, and the block under it must be deleted. The first provides the continuation of boundary layer which comes from the "key brick" block used for blocking around valve, and it is showed in light-blue in Fig. 4.77a. The second provides the O-grid in final part of intake duct, and it is showed in orange in Fig. 4.77a. The fact there are two O-grid block is due to mesh cells: the nearest O-grid to the wall can have a maximum of 2-3 cells since its spacing is strictly related to spacing of conical space between valve-side and valve-seat. Since the valve closure cannot be simulate as it happens in reality (no cells volume are not allowed in CFD), but with `attachDetach` topology modifier, at least one cell always remain in this very tight conical space. Thus it is not worth to use a lot of cells because it will limit Courant-Friedrichs-Lewy (CFL) number, in the further simulation, when the valves are closest to the their closure.

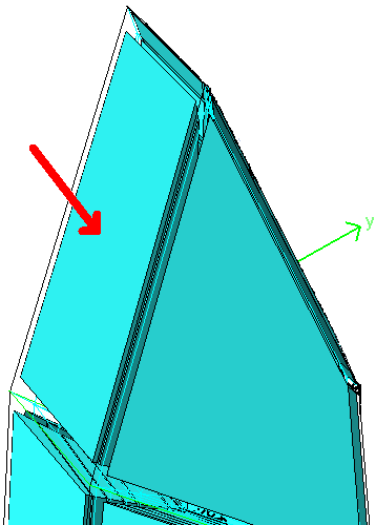


Figure 4.81. Red arrow shows the block that not provides boundary layer if was extruded

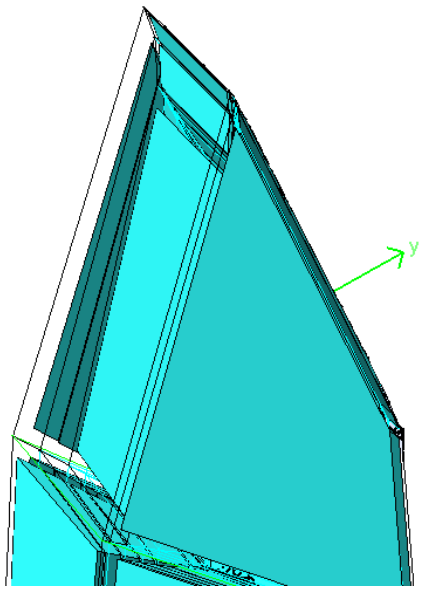
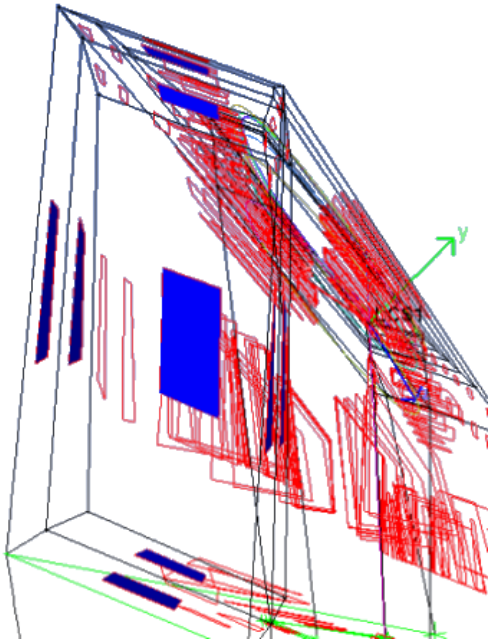


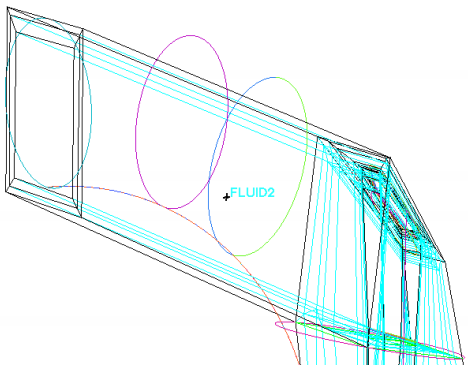
Figure 4.82. structure after block deletion

5. Construction of final part of intakeDuct; Therefore, in order to obtain the final block part of intakeDuct two block must be deleted to create O-grid structure. The result of deletion is shown in Fig.4.82. Then blocks have been generated

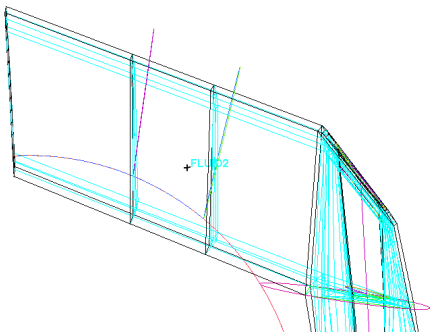
with extrusion of faces, shown in Fig. 4.83a, using "extrusion along curve" tool. Afterwards, a split (Fig. 4.83c) has been applied to fit better the curvature of duct.



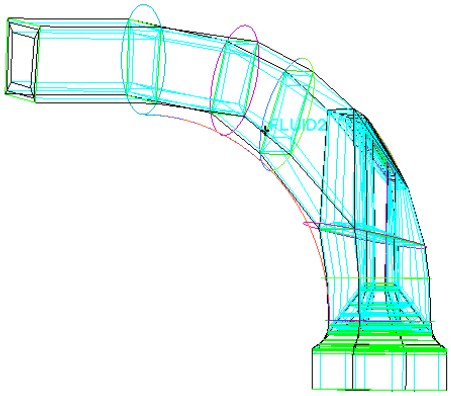
(a) *Selection of faces to extrude.*



(b) *Extrusion along intake duct.*



(c) *Split of block.*



(d) *Final block and association of edge to geometry.*

Figure 4.83. Steps to obtain final blocks of duct

6. Mirroring of intake-block part to obtain the exhaust-block part. Since intake and exhaust are geometrically identical and symmetric, as shown in Fig. 4.48a, it is worthwhile to use the same blocks of intake and the geometry created during adjustment as well. Thus either geometry and the blocks need to be mirrored on his plane of symmetry. The operation can be made in two steps in ICEM: Mirror geometry tool and Rotate geometry tool (under **Geometry-Transform geometry**). Then Mirror blocks tool and Rotate blocks (under **Blocking-Transform blocks**) tool.

   Since a lot of adjustments have been done on the geometry to provide the construction of blocking and edges association, the intake geometry is mirrored too. The mirror geometry tool wants in input the geometry part to mirror (point, curves and surfaces) and it is necessary specify:

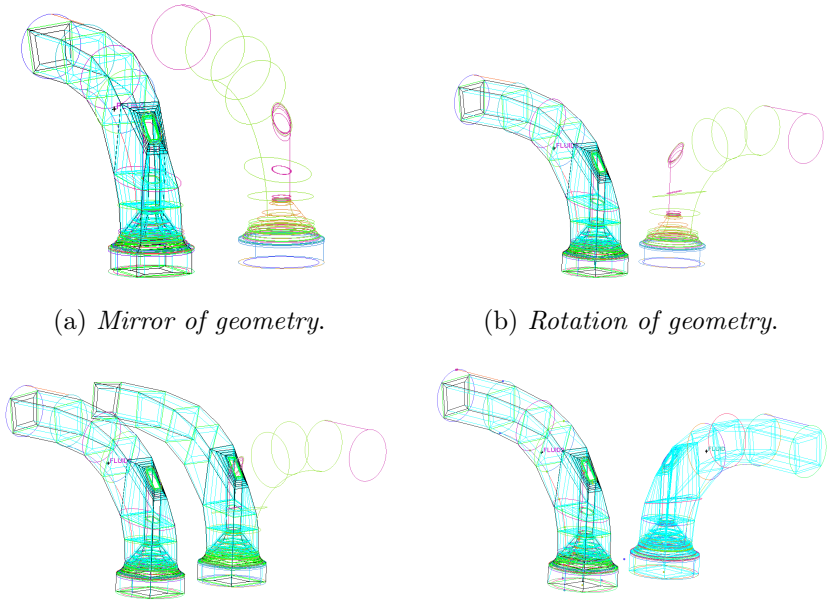
   - "copy" because it needs one geometry more for exhaust;
   - "increment parts" for all parts belonging to intake duct. Once the mirror is done, the "increment parts" option produce in the tree parts a new part whose name is the old part plus _0 (e.g inValve-stem becomes inValve-Stem_0). Thus each new part must be changed with a name for exhaust;
   - The axis perpendicular to plane of reflection;
   - The point of reflection, on which the plane of reflection must pass through;

   Then the new geometry part shown in Fig. 4.84a must be selected to be rotated (4.84b) and it needs to specify:

   - The axis or vector around which the rotation occurs;
   - The point on which the axis of rotation must pass through;

   .

   The same has been performed on block as shown in Figs.4.84c 4.84d. However it is not allowed to generate an "increment part" for block. Thus the new blocks own the same association of native blocks. Thus old association must be deleted and replaced with new ones. The dissociation from geometry is understandable from the light-blue colour of all edges in Fig. 4.84d.



(a) *Mirror of geometry.*

(b) *Rotation of geometry.*

(c) *Mirror of blocks.*

(d) *Rotate of geometry and dissociation from geometry.*

Figure 4.84. Operation of mirror and rotation on geometry and blocks

7. Construction of intake/exhaust plenum and merging with ducts. intake and exhaust plenum own a different. The difference lies on height (Fig. 4.85) and bottom patches of tanks as shown in Fig. 4.86 and Fig. 4.87.

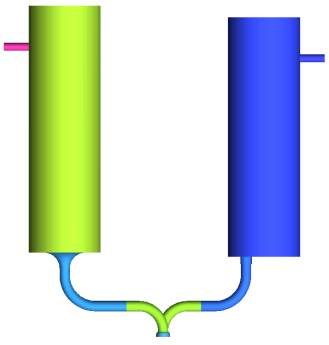


Figure 4.85. Lateral view of intake (green) and exhaust (blue) plenum

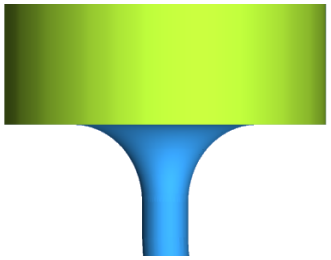


Figure 4.86. Lateral view of intake bottom-tank

Figure 4.87. Lateral view of exhaust bottom-tank

Nevertheless their blocking has been made independently, because are different, their generation is similar. They are two simple cylinder-cylinder intersection. thus an O-grid crosses another O-grid in both cases. the result is shown in Fig. 4.88

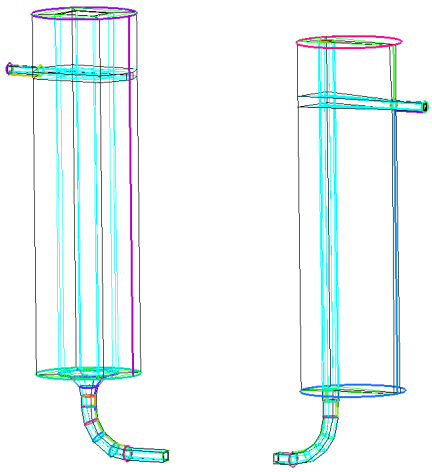


Figure 4.88. shows the block of intake and exhaust plenum

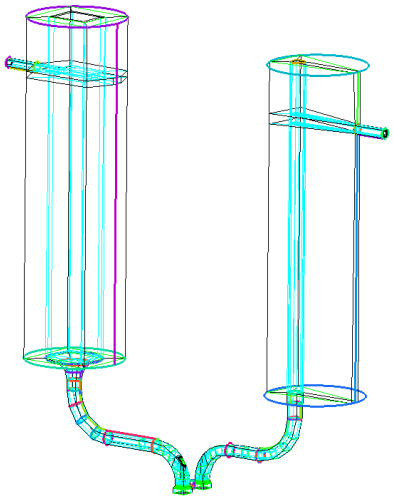


Figure 4.89. shows the merging between plenum and ducts

The blocks of plenum have not been generated in the same project of intake and exhaust duct but in another one. Afterwards, from the project of ducts, the blocking file of only the plenum has been open and merged with the "root" Blocking. ICEM does not merge the blocks immediately but generates a sub-topology corresponding to the new block. "Topology" is displayed under Blocking tree and give to the user the information about "root" topology and sub-topology. Right clicking on sub-topo gives many option. One of this allows to merge sub-topo with "root" topology. After that the two topologies are displayed together in ICEM window. Thus now it is possible to merge blocks at their meeting point, merging vertexes to corresponding vertices. The final block structure is shown in Fig. 4.89

Blocking definition and adjustment for cylinder

Since the cylinder has the geometry of Fig. 4.48c more than one O-grid block are necessary, either for boundary layer near liner patch either for the three cylindrical non-conformal interfaces. The synthetic block approach use is shown in Fig. 4.90

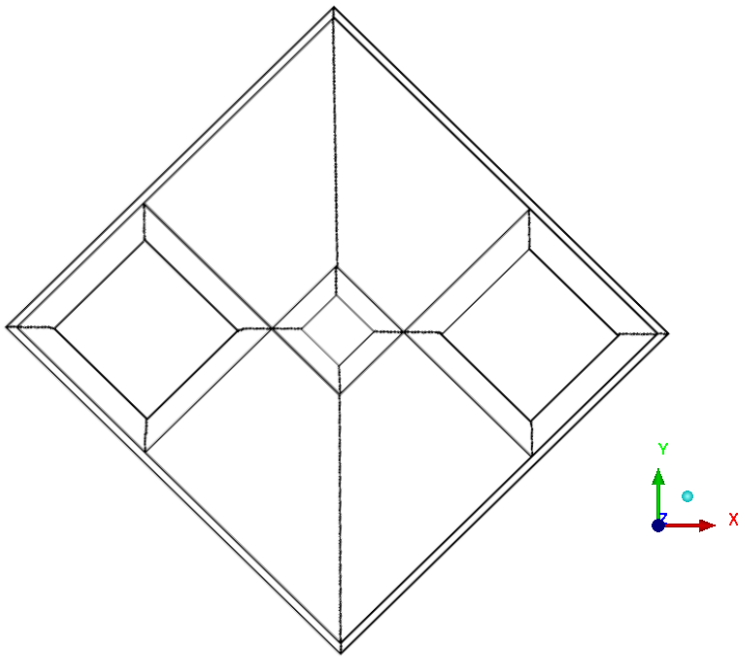


Figure 4.90. Section view of block structure adopted for cylinder

After the creation of the first block and first O-grid, it has noticed out that the only way to obtain blocks of Fig. 4.90 is rotate the block just created of 45° degrees around the z axis. ICEM block are automatically created oriented with global coordinate system, thus a block or geometry rotation is strictly necessary.

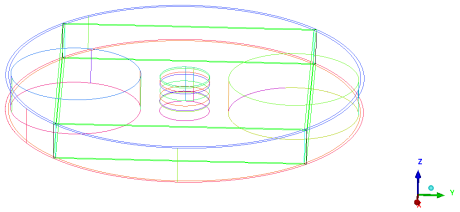


Figure 4.91. First O-grid creation

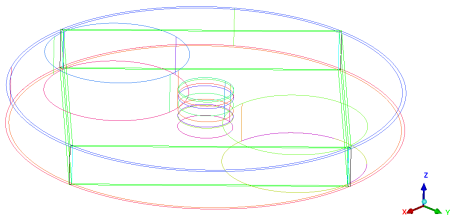
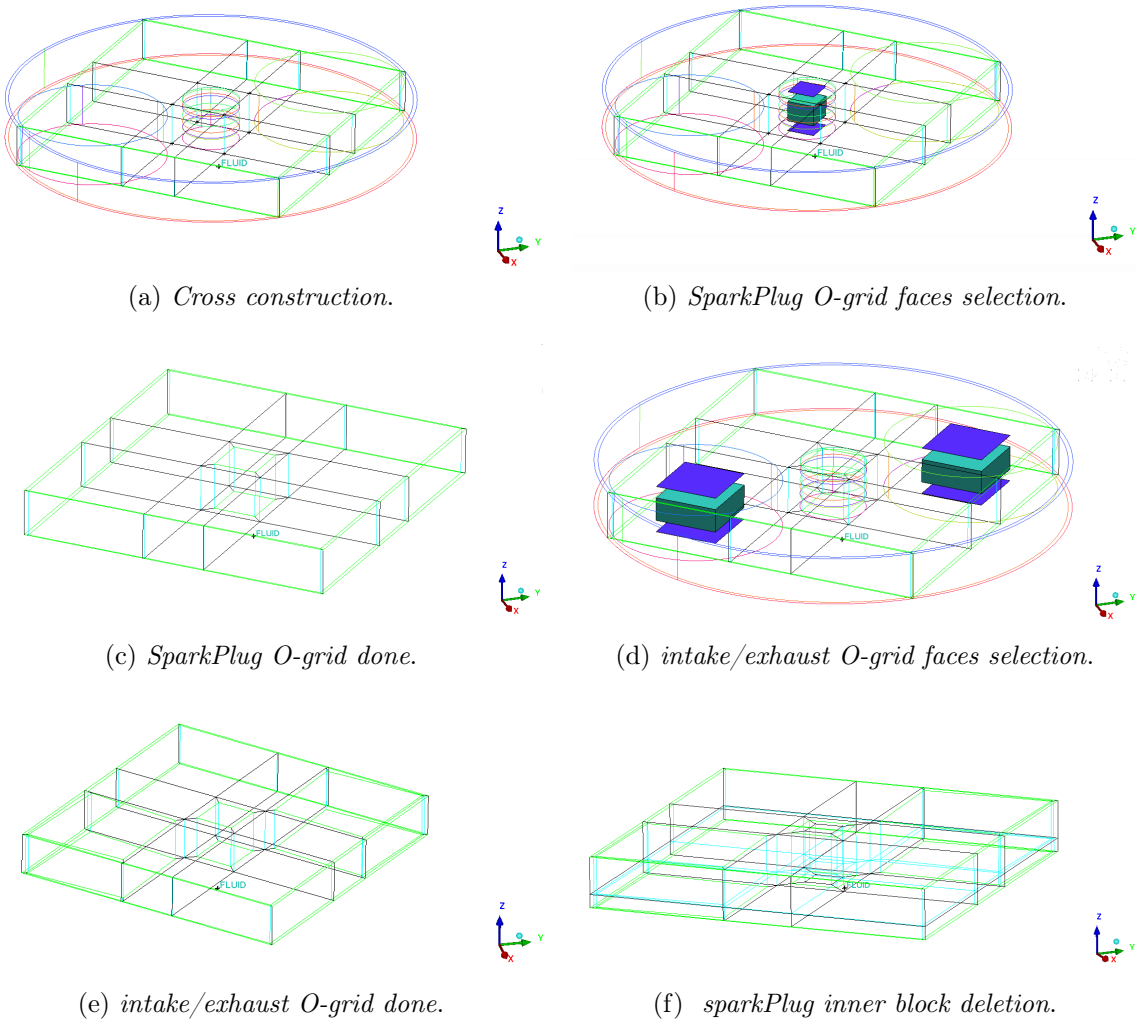


Figure 4.92. Block rotation

Then a cross has been built centered on sparkPlug. It has been generated splitting perpendicular the block twice in both direction. Afterward three O-grid have been built, two for AMI and one for static non-conformal interface between cylinder and

spark plug, as shown in Fig. 4.93.



(a) *Cross construction.* (b) *SparkPlug O-grid faces selection.*

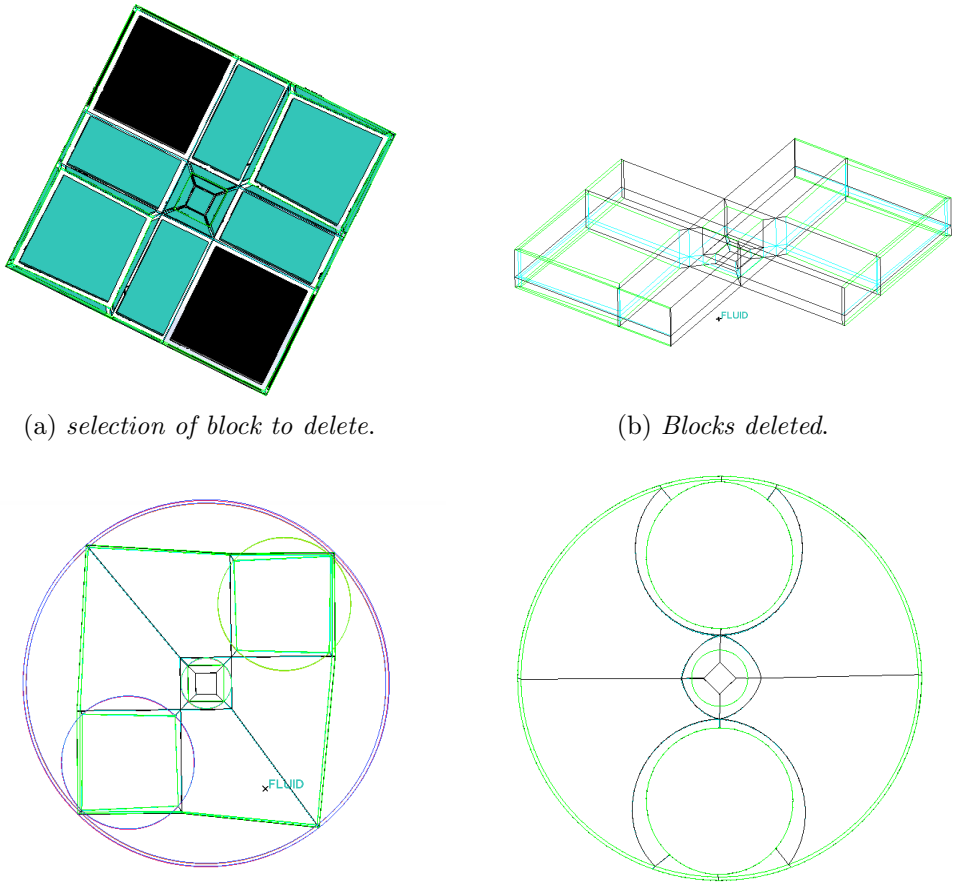
(c) *SparkPlug O-grid done.* (d) *intake/exhaust O-grid faces selection.*

(e) *intake/exhaust O-grid done.* (f) *sparkPlug inner block deletion.*

Figure 4.93. Steps for the creation of O-grid

The inner block of Fig. 4.93f must be deleted to contain further the mesh of sparkPlug. However not all the block in his own height must be deleted as can possible see from geometry in Fig 4.48c and 4.48d. Thus all blocks are splitted along z direction and one more O-grid as been built inside the inner block. Then the top part of this block has been deleted. Fig .4.94a shown also some blocks in black. These blocks have to be removed to have the block of Fig. 4.90 and free vertices has been merged as well. Finally the blocks are adjusted, moving vertices and applying "automatic split edge"

(as seen in Sec. 4.3.1), to fit the geometry (Fig. 4.94d).



(a) *selection of block to delete.*

(b) *Blocks deleted.*

(c) *Blocks after vertexes merging.*

(d) *Blocks after association and automatic edge splitting.*

Figure 4.94. Steps of deletion and association for final cylinder structure

Blocking definition and adjustment for sparkPlug

As can possible see in Fig. 4.48d, spark Plug geometry is really complex to hexa-meshing: his top part is a cylinder-like geometry while the bottom part is a smooth square prism. Whilst the top part (central electrode) wants O-grid structure , the bottom part (ground electrode) does not want it, thus a compromise between O-grid structure and classical hexa blocking must be made. Moreover the fact that the ground electrode is tangent to the cylindrical part of spark makes harder to mesh it. The following procedure is repeatable on other spark geometry since the huge part of it does

not change remarkably; only the distance between electrode and the volume between thread and central electrode can change.

The blocking procedure starts from the volume between external thread and central electrode, where the presence of an O-grid structure is necessary (Fig. 4.96).

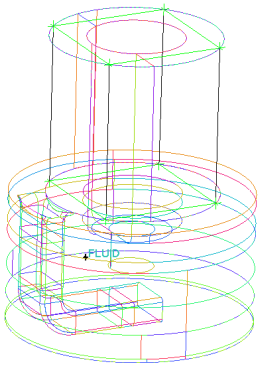


Figure 4.95. First block top part of spark Plug

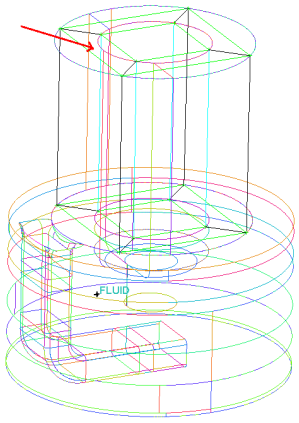


Figure 4.96. O.grid on the top part of spark Plug

However as said previously the O-grid must be adjust to fit the complex geometry of spark plug. For this reason the block pointed out in Fig. 4.96 has been deleted to allow the creation of Quarter O-grid, the same use for valve-side blocks. After deletion, the old internal face become an external face thus a new block can be created from it with a simple extrusion as shown in Fig. 4.114a

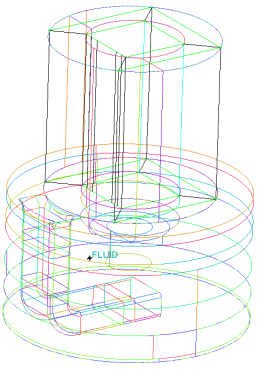


Figure 4.97. Deletion of block

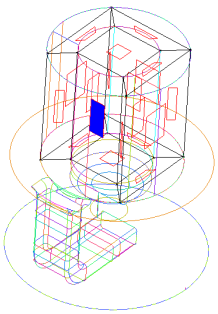


Figure 4.98. Selection of face to extrude

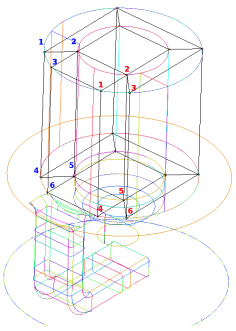


Figure 4.99. Extrusion of face

In this way Quarter O-grid can be generate as seen in Fig. 4.68 and result is shown in Fig.4.100.

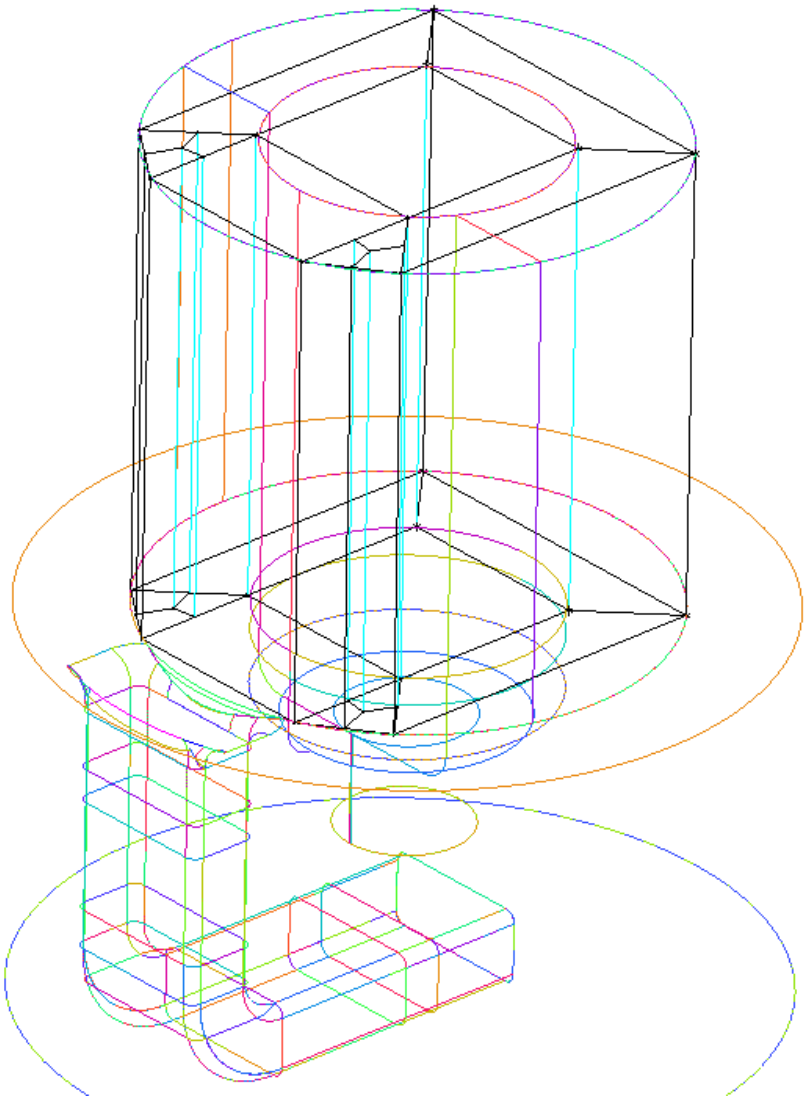


Figure 4.100. Quarter O-grid on the top part of spark plug

This structure will be kept as it is to the bottom surface, however in the inner block two more O-grid will be done to fit the central electrode geometry. Thus bottom blocks' faces has been extruded (Fig. 4.101) and 3 split has been performed according to geometry (Fig. 4.102).

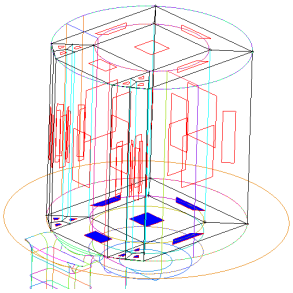


Figure 4.101. Selection of bottom faces to extrude

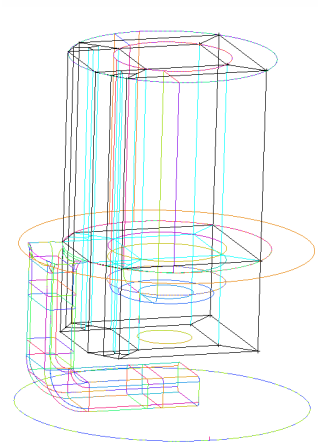


Figure 4.102. Extrusion of faces

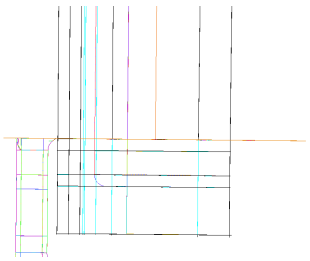


Figure 4.103. Splitting of block in 4 sub-blocks

The block which represent the head of the central electrode has been modified with two O-grid and afterward the new faces' structure of the most bottom block has been extruded to the bottom surface and four more split has been done.

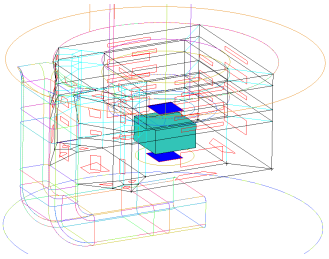


Figure 4.104. Selection of block and faces for first O-grid for central electrode

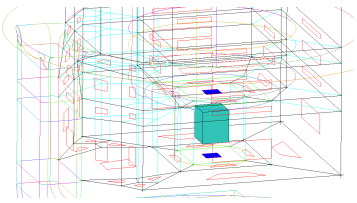


Figure 4.105. Selection of block and faces for second O-grid for central electrode

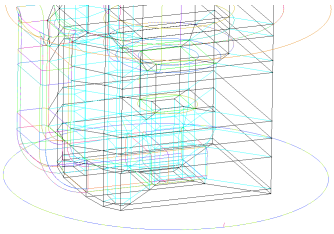


Figure 4.106. Extrusion of faces from bottom of central electrode to the lower surface of geometry, and further splitting of block in 4 sub-blocks

The central block structure owns some blocks that must be deleted because inner part of geometry is void. Thus inner blocks from the top face to the bottom face of central electrode has been deleted as shown in Fig. 4.114a.

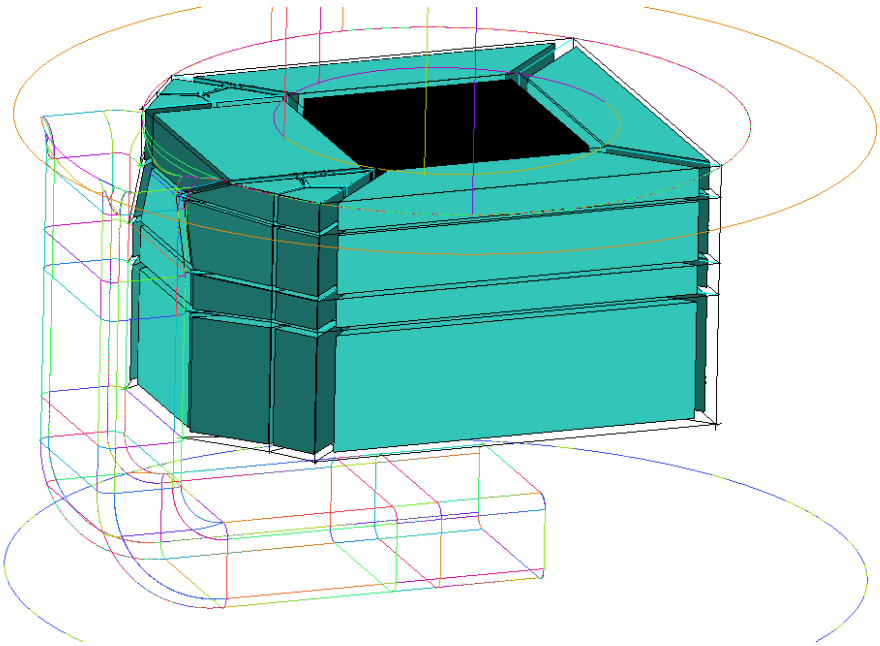


Figure 4.107. Deletion of inner blocks

In order to obtain the Most external part of blocking structure extrusion from all external faces must be performed. Then external vertexes has been merged to average.

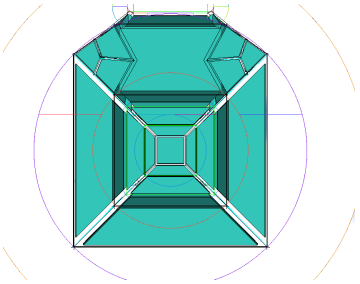


Figure 4.108. Top view of block structure

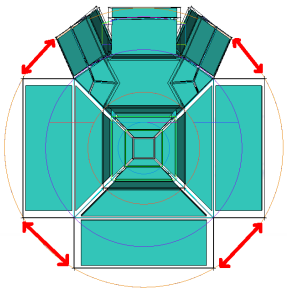


Figure 4.109. Extrusion of external faces

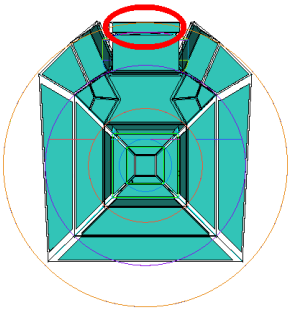


Figure 4.110. Merging vertexes of adjacent blocks

The blocks, which has just been merged, needs another split in radial direction to merge the small blocks pointed out in Fig.4.110. The result is shown in Fig. 4.112.

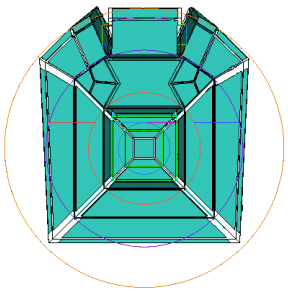


Figure 4.111. Radial splitting blocks

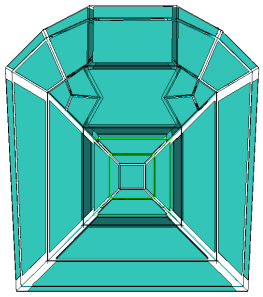


Figure 4.112. Final merging of vertexes

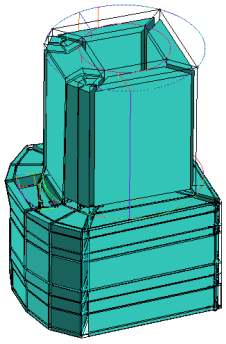
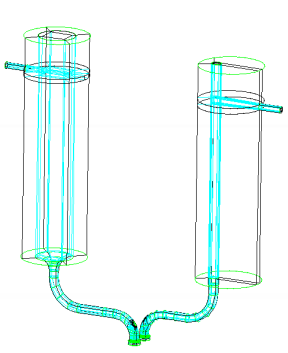


Figure 4.113. Final block structure of sparkPlug

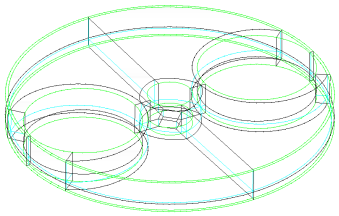
To obtain final block structure, the blocks which occupies the geometry of ground electrode need to be removed since it is a solid region and not a fluid region. After these blocks deletion the final blocks is that of Fig. 4.113

Mesh definition: check and adjustment

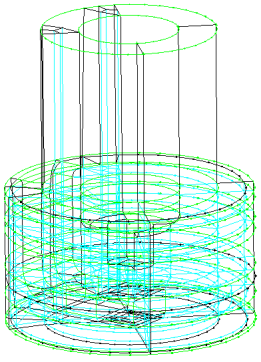
After the application to blocks the "automatic linear" splitting tools at each part (cylinder, intake/exhaust, sparkPlug) and after the choice of the suitable spacing near wall patches a "smooth mesh globally has been applied " to reach a good value of skewness.



(a) *intake/exhaust blocks.*

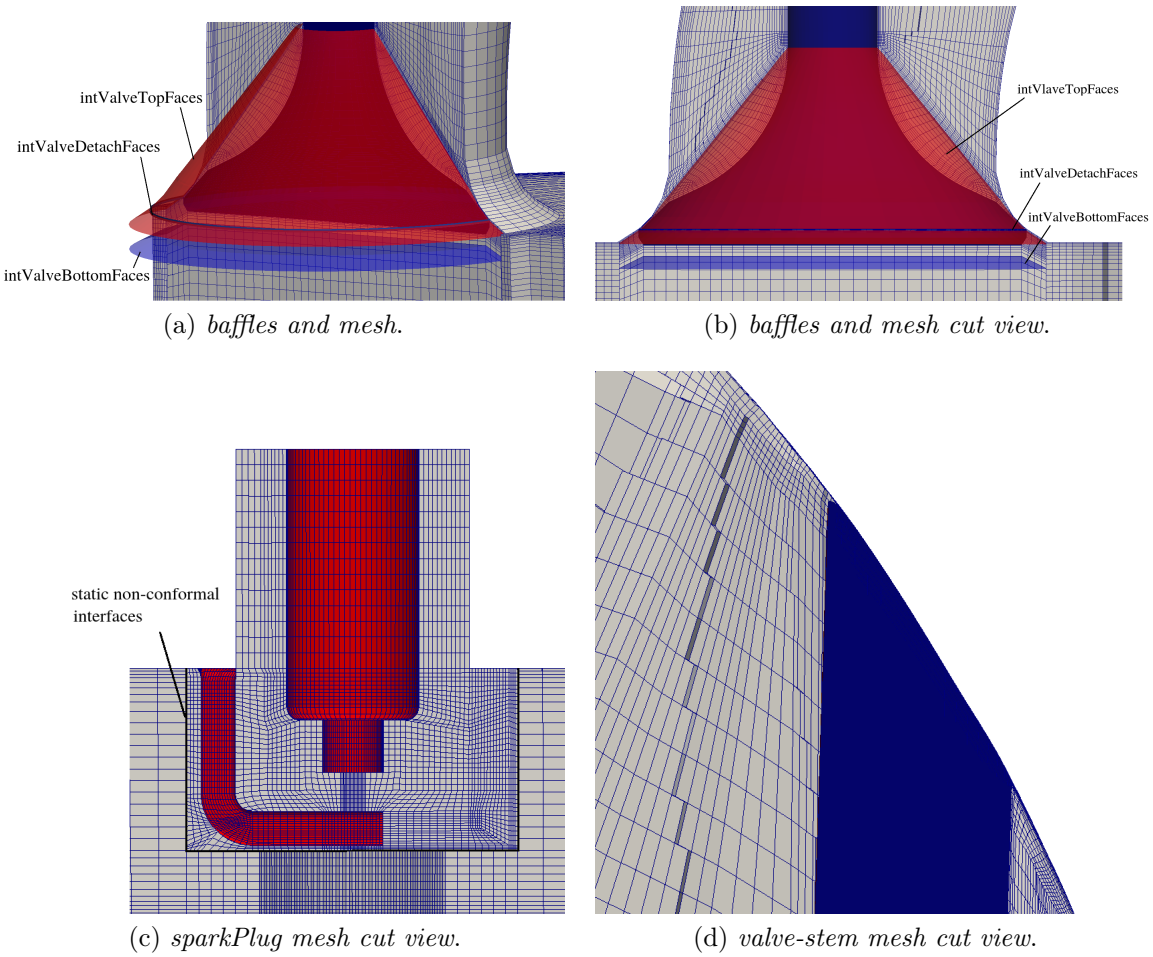


(b) *cylinder blocks.*



(c) *sparkPlug Blocks.*

Figure 4.114. Shows the final blocks for each part



(a) *baffles and mesh.*

(b) *baffles and mesh cut view.*

(c) *sparkPlug mesh cut view.*

(d) *valve-stem mesh cut view.*

Figure 4.115. Shows the final mesh for sparkPlug and near valve region

More attention must be given to intake/exhaust duct, especially near baffles. After unstruct mesh smoothing it is possible to extract the mesh part corresponding to baffles' geometry created at the beginning:

- From tree parts the user choose the baffles to display. In this way on ICEM window will appear only that mesh part.
- ICEM allows to "save only the visible mesh". Thus this has been done on inVale-detach, exhValve-detach and inValve-topfaces, exhValve-topFaces.

For inValve-bottomfaces and exhValve-bottomFaces a different approach has been used because no-one baffles have been created:

- Creation of subset from mesh tree, for sliding-intValve patch and sliding-exhValve patch (Fig. 4.118);

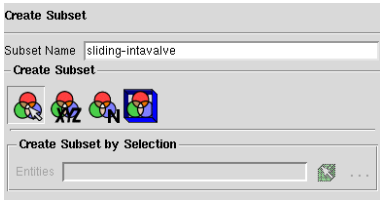


Figure 4.116. Create subset window

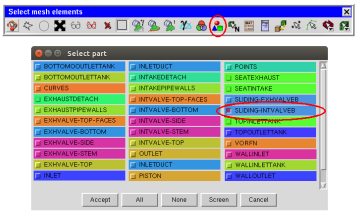


Figure 4.117. Create subset from parts

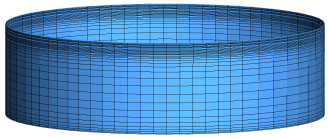


Figure 4.118. Subset created

- Removal of layers from the bottom and the top of subset (Fig. 4.114a). From the top only two layers has been removed (between valve seat and intakeTopFaces there are only a two layer cell). The number of layers removed from bottom depends on how many layer we want between valve-bottom and intake-bottomFaces (Fig. 4.114a).

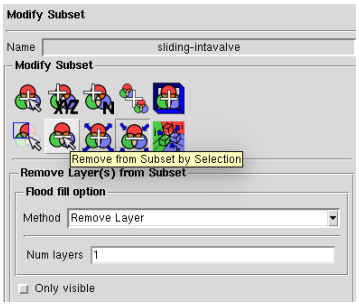


Figure 4.119. Remove layers from subset window

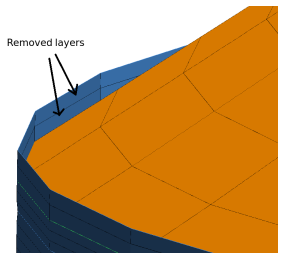


Figure 4.120. Remove layer from top

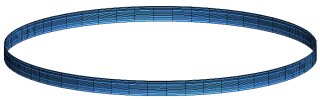


Figure 4.121. Final subset

- Creation of subset for intValve-bottom;

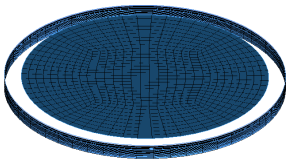


Figure 4.122. Creation of valve-bottom subset

- Modification of intValve-bottom subset. Choosing "add layer to subset" with add layer method and " also volume" option activated, the user add 1 layer per time until cells reach the bottom of sliding-intValve subset created previously (Fig. 4.114a).

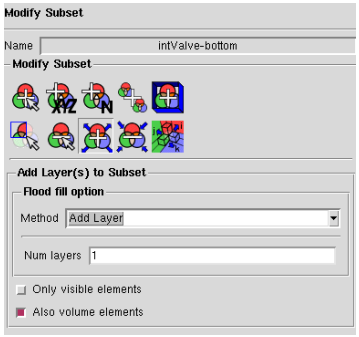


Figure 4.123. Add layer to subset window

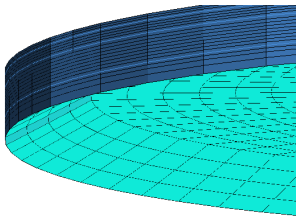


Figure 4.124. Cells added to subset

- Saving of the "only visible mesh";

- Opening of another project containing only the previous mesh subset. The user has to apply a checkMesh activating the control on uncovered faces. During check process a warning message will appear because the bottom cells are uncovered. Thus ICEM asks if the user want to cover them, and it allows to specify the name of the patch that will cover the fluid cells. This new patch will appear in tree part. This new patch will be intValve-bottomFaces and exhValve-bottomFaces (Fig. 4.114a).

- Displaying only intValve-bottomFaces and exhValve-bottomFaces save it with "save only the visible mesh".

- Creation of exhaust-movingCells and intake-movingCells. These subset represent all the faces containing (Fig. 4.125) the moving cellSet (Fig. 4.126). Thus open another project in ICEM and open intakeToFaces.uns ,intake-bottomFaces.uns ans the sliding-exhvalve.uns creating previously from subset. The same thing for exhaust. Then save it as exhaust-movingCells.uns and intake-movingCells.uns.

- Finally creation of .stl file. For each file the user has to create a stl file in order to create faceSet and cellSet with topoSet tool in OpenFOAM. Thus under **File-export mesh-write STL file** the stl file will be created.

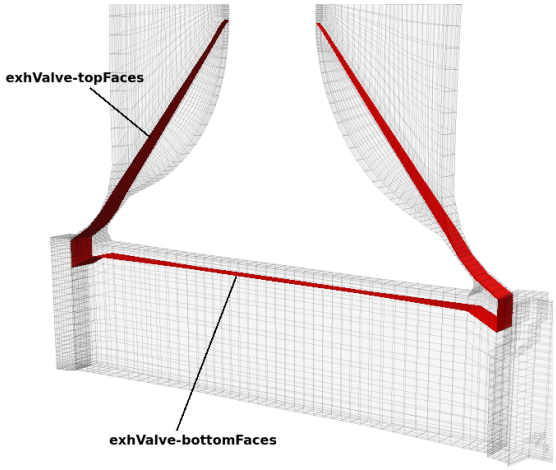


Figure 4.125. faceSet involved in moving Mesh

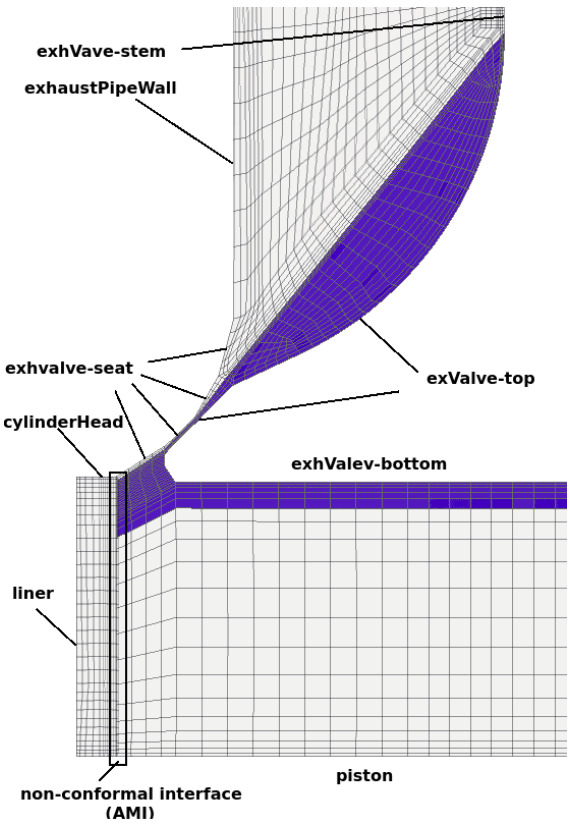


Figure 4.126. CellSet involved in moving Mesh

4.4 Interfacing ICEM mesh with OpenFOAM

4.4.1 Mesh export and mesh conversion in OpenFOAM

For both cases, once the meshes fulfil the quality constrains, the .uns file has to be converted to .msh file and the user have to assign to all patches the wall boundary condition (BC) type. OpenFOAM accepts a wide variety of unstructured mesh file from the most common solvers such as fluent, kiva, gambit, cfx, STAR-CD. Since i decide to export .uns file to fluent mesh file .msh i will use a fluent mesh converter:

- `fluentMeshToFoam` is an old code implemented in OpenFOAM and handles 2D Fluent meshes but struggles with some 3D meshes, especially when there are baffles patches. Thus it is possible use it with flat-top cylinder case but not with TCC case.
- `fluent3dMeshToFoam` is newer. Can handle only 3D meshes, but it does this much better than the old one.

This tool can be used together with some options. The most common used is the scale option. Since OpenFOAM is based on international system of units, it accepts mesh in meters. Thus it allows to scale the mesh if it was realize on a geometry not previous scale to the right magnitude.

4.4.2 mergeMeshes tool

TCC is made up of three pieces: intake duct & plenum, exhaust duct & plenum, sparkPlug, cylinder. All the pieces need to be merge in a unique mesh file under `polyMesh`. This tool place the mesh definition for each piece in the same destination folder but does not merge them physically (stitched) The tools is used n times where n is the number of pieces: each time a masterCase and a slaveCase are chosen. The slave one will be merge in the master one.

4.4.3 mergeOrSplitBaffles tool

As seen in Sec. 4.3.2 in mesh adjustment, Intake and exhaust duct own in their inner volume an internal patch used to create the STL geometry surface that identifies the face on which the `layerAdditionalRemoval` topology modifier has to be applied on. Although baffles are useful to create the STL file to invoke in `topoSetDict` they are useless during the simulation and create problems for dynamic mesh Since the domain is divided in more region instead of one (when valves are opened). Thus this tool allows the baffles removal to guarantee the right definition of fluid dynamic domain. The tool is applied directly on the case with whole mesh.

4.4.4 projectPatchPoints tool

This tool is used to prevent errors during stitch procedure. Since stitching is based on an algorithm that starts from the normal vector to a cell face, if faces belonging to the same patch own different directions relative to other patch to stitch with, the mesh cannot be stitched. Thus `projectPatchPoints` allows to project the point's of the slave patch onto surface identified by master patch. The tool is applied choosing a master patch on which point of slave patch are projected, the information are invoked in `projectPatchPointsDict`. The `projectPatchPoints` tool has been applied:

- on sparkPlug for the further stitch with the correspondent patch on cylinder;
- on intake/exhaust non-conformal interface in order to provide a correct correspondence between mesh faces during piston motion.

4.4.5 stitchAndSplitMesh tool

Finally `sticthAndSplitMesh`,which rely on slidinginterface toplogy modifier, is applied between sparkPlug and cylinder patches up to a tolerance, invoked in file `toleranceDict`,

which differs from case to case: the finer is mesh the bigger can be the tolerance, seen as the distance between two points belonging to different patches.

4.4.6 topoSet tool

Both case study build their topological changes on `faceSet` and `cellSet` created by `topoSet` tool. `topoSet` invokes `topoSetDict`. In the first part of `topoSetDict` label are assigned to STL file

```
exhValveTopFacesSTL     "constant/triSurface/exhValve topFaces.stl";
exhValveBottomFacesSTL "constant/triSurface/exhValve bottomFaces.stl";
exhValveDetachFacesSTL "constant/triSurface/exhValve detach.stl";

intakeTopFacesSTL     "constant/triSurface/intValve topFaces.stl";
intakeBottomFacesSTL "constant/triSurface/intValve bottomFaces.stl";
intakeDetachFacesSTL "constant/triSurface/intValve detach.stl";

exhValveMovingCellsSTL "constant/triSurface/exhaust movingCells.stl";
inValveMovingCellsSTL "constant/triSurface/intake movingCells.stl";
```

Then it is possible to specify a series of tolerance to invoke in the generation of set, case by case:

```
tolerance_1 1e 5;
tolerance_2 1e 5;
tolerance_3 1e 5;
```

Afterwards for each .stl file invoked in `sourceInfo` a `type` of faceSet is created, up to a specific tolerance, according to the source file (e.g if set originates from a stl geometry `surfaceToFace` is used, otherwise if it originates from a patch `patchLayersToFace` is used). For valves the procedure has been done on top-Faces, bottom-Faces, detach-Faces, for both intake and exhaust basing on the baffles created previously in ICEM.

```
    //  exhValve: top Faces
    {
        name   exhValve topFaces;
        type   faceSet;
        action new;
        source surfaceToFace;
        sourceInfo
        {
            file  $exhValveTopFacesSTL;
            nearDist $tolerance_1;
        }
    }

    //  exhValve: bottom Faces
```

```
{
    name    exhValve bottomFaces;
    type    faceSet;
    action  new;

    source  surfaceToFace;
    sourceInfo
    {
        file  $exhValveBottomFacesSTL;
        nearDist $tolerance_2;
    }
}

//  exhValve: detach Faces
{
    name    exhValve detachFaces;
    type    faceSet;
    action  new;
    source  surfaceToFace;
    sourceInfo
    {
        file  $exhValveDetachFacesSTL;
        nearDist $tolerance_3;
    }
}
```

For piston faces the procedure has been done on same way on piston, intake-piston and exhaust-piston patches:

```
// piston layer AR
{
    name    pistonFaces;
    type    faceSet;
    action  new;
    source patchLayersToFace;
    sourceInfo
    {
        patch piston;
        nLayers $pistonLayers;
    }
}
```

The same has been done in order to create cellSet. cellSet gives the information about cells involved in the motion. Thus cellSet is created for both intake and exhaust valve:

```
//  exhValve: movingCells
```

```
{
    name    exhValve movingCells;
    type    cellSet;
    action new;
    source surfaceToCell;
    sourceInfo
    {
        file $exhValveMovingCellsSTL;
        useSurfaceOrientation true; //false;
        outsidePoints
        (
            (0.035 0.0006 0.015)
            (0.045 0.0006 0.001)
            (40e3 0.6e3 4.19e3)
            (42e3 0.6e3 1e3)
        );
        includeCut      false; //true;
        includeInside   true;
        includeOutside  false;
        nearDistance     1;
        curvature        100;
    }
}
```

Also for piston motion a cellSet has to be defined, usually this coincide with the boundary layer on piston patch whose layering must keep unchanged not applying no-one kind of topology modifier on it. Thus a number of `pistonLayers` has to be defined. This value will be invoked in `sourceInfo` to create the pistonCells:

```
{
    name    pistonCells;
    type    cellSet;
    action  new;
    source patchLayersToCell;
    sourceInfo
    {
       patch piston;
       nLayers $pistonLayers;
    }
}
```

However, comparing with faceSet type, the `source` changes since moving cellSet has been built from a surface including those cells, and piston cellSet has been built beginning from the patch itself to a specific number of layers.

It is important underlying that piston faceSet and cellSet ave been built both for flat-top cylinder head and TCC, while intake/exhaust valve faceSet and cellSet have

been built only for TCC.

4.4.7 createPatchDict-AMI tool

In TCC case the moving non-conformal interface (slidingIntakeA-B and slidinExhaustA-B) has not been handled with sliding interface but with AMI. Since OpenFOAM needs information about patches gather up in AMI, a createPatchDict-AMI is requested. This gives information about each patch and its neighbour one:

```
{
    name slidingIntakeA;
    patchInfo
    {
        type cyclicAMI;
        inGroups        1(cyclicAMI);
        matchTolerance  0.01;
        transform       noOrdering;
        neighbourPatch  slidingIntakeB;
    }

    constructFrom patches;
    patches (slidingIntakeA1);
}

{
    name slidingIntakeB;
    patchInfo
    {
        type cyclicAMI;
        inGroups        1(cyclicAMI);
        matchTolerance  0.01;
        transform       noOrdering;
        neighbourPatch  slidingIntakeA;
    }
    constructFrom patches;
    patches (slidingIntakeB1);
}
```

Chapter 5

Cases set-up and results

In this section the set-up of two cases has been dealt with: for the Flat-top cylinder head with a fixed, axis-centered valve more than one full cycle have been performed to analyse axial velocity and the rms velocity, instead for the TCC case only the piston compression and expansion have been performed. Finally a comparison between experimental results, DLRM and LES has been performed.

5.1 Flat-top cylinder head with a fixed, axis-centered valve

5.1.1 Experimental set-up

Specification of the geometry for the case studied [16] as well as a list of main functioning parameter has been represented in Fig. 4.14. The piston has a bore diameter of 75 mm, stroke is 60 mm and clearance height is 30 mm from the cylinder head at the Top Dead Center (TDC); the geometric compression ratio is 3. The poppet valve is coaxial with respect to the piston axis and it is static; the valve angle is 30° with respect to the cylinder axis and the width of the uniform valve gap is 4 mm.
Piston motion is purely harmonic with a frequency of 200 RPM; the piston has a mean speed $\bar{\mathbf{U}}_p = 0.4\, m/s$ and it reaches its maximum speed at 90° and 270° CA degrees ATDC; as a consequence, the engine Reynolds number (considering air as a working fluid) is $Re = \rho \bar{\mathbf{U}}_p D/\mu \sim 2000$.
The experimental data used for a comparison with CFD simulation has been taken from [16]. The experiments were carried out with air at environmental conditions (1 atm and 293 K). Laser-Doppler anemometry was used to measure the axial and azimuthal velocity components at 36° , 90° , 144° CA during the intake stroke and at 270° CA during the exhaust stroke at points located on planes at 10 mm, 20 mm and 30 mm below the cylinder head in the axial direction. The mean and rms values of the axial velocity were obtained by averaging over 100 samples within a 10° CA interval

and five independent sets of measurements were taken to check the reproducibility of the results [16]. The reported errors were smaller than ±3% for the mean and ±5% for the rms velocity; in regions with steep gradients the error in the mean field increases to up to 10 % for the mean and to 20 %for the rms values.

5.1.2 Numerical set-up and methodology

The geometry has been discretized with a pure unstructured hexahedral mesh as shown in Sec. 4.3.1. The cylinder inlet section is connected to a plenum (Fig. 5.1) whose volume is about 11 times the cylinder volume since cylinder volume at BDC is about $0.0004\,m^3$ and tank volume is about $0.0043\,m^3$ ($Tank - Diameter = 0.166\,m$, $Tank - height = 0.2\,m$)

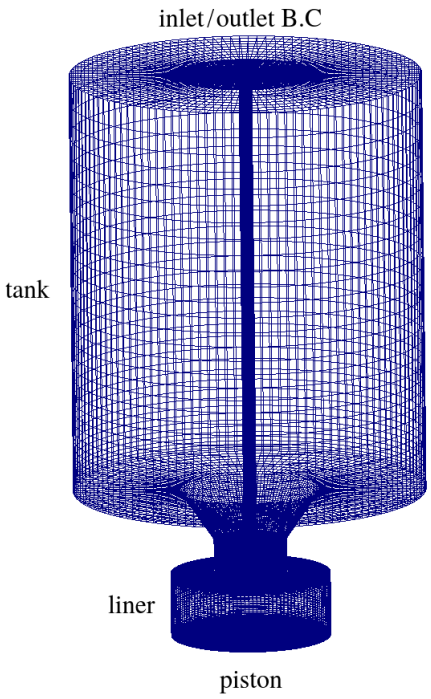


Figure 5.1. Case set-up and boundary conditions for the apparatus experimentally studied used by Morse, [16]. The outlet boundary condition provides a non-reflecting outflow condition, with specified inflow for the case of return flow occurring during the intake stroke.

The purpose of the plenum was to avoid reflections from the inlet boundary, to improve the accuracy of the results and the stability of the simulation.

Thanks to the hybrid RANS/LES turbulence model applied, a very coarse mesh, whose resolution was ranging from 1.400 K cells at the BDC to 800 K cells at the TDC has been used for simulations (no plenum cells are taken into account). In previous LES study in [19] the grid had about 4 million cells. The strategy chosen for dynamic mesh handling is based on dynamic layer addition/removal: the cells are stretched/squash up to a value of thickness during piston motion; then, beyond this value, a layer is added/removed if there expansion or compression respectively.

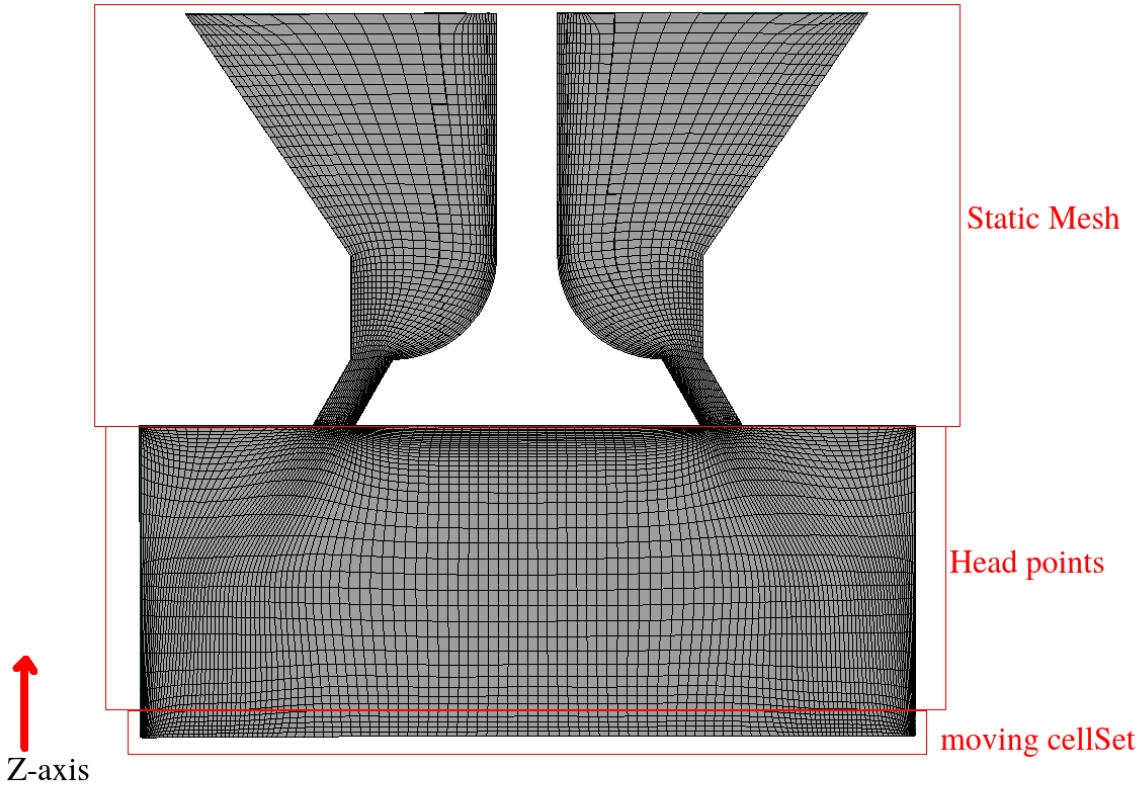


Figure 5.2. Cut view of the Finite Volume grid used for the simulations. The whole mesh had about 1 million hexahedral elements at TDC, including the plenum (not shown).

The computational grid could be divided into three sub-regions, where points are characterized by a different motion law (see Fig. 5.2) In particular, points above the cylinder head were static and they belong to a sub-region named “static mesh”.

Similarly, points located below the cylinder head ("head points", Fig. 5.2) were not moving to preserve the cell quality in the region between valve seat and cylinder: this region goes from cylinderHead patch to the first layer of cell not belonging to moving cellSet (usually 4/5 layers) defined in topoSet as shown in Sec. 4.4.6 for piston patch. Finally, points belonging to moving cellSet generated from piston patch, move axially to account for the moving boundary (piston). Assuming the cylinder axis aligned to the z axis of the global reference frame of Fig. 5.2, the point velocity $\mathbf{u}_p$ is calculated as follows:

$$\mathbf{u}_p(\mathbf{x}_p) = \mathbf{U}_p(t)\frac{z_{max} - z_p}{z_{max} - z_{piston}} \tag{5.1}$$

where :

$$\mathbf{U}_p(t) = S/2\sin(\omega t)\mathbf{i}_c \tag{5.2}$$

is the piston velocity, S is the piston stroke, $\mathbf{i}_c$ is the unit vector parallel to the cylinder axis, $\mathbf{x}_p = (x_p, y_p, z_p)$ is the point position, z_{max} is the z-coordinate of the farthest moving point from the cylinder and z_{piston} is the z-coordinate of the piston.

Case set-up

The boundary and initial condition for involved physical quantities are presented briefly in Tab. 5.1

	inlet	/tank	cylinderHead, liner, /valve	piston	internalField
k	inletOutlet	zG	wF	wF	$0.01\ m^2/s^2$
ω	inletOutlet	zG	wF	wF	$400\ s^{-1}$
μ_t	calculated	zG	wF	wF	$1e-8\ kg/(ms)$
P	totalPressure	zG	zG	zG	$101325\ Pa$
T	inletOutlet TotalTemperature	zG	zG	zG	$293\ K$
U	pressureInlet OutletVelocity	$0\ m/s$	$0\ m/s$	mWV	$0\ m/s$

Table 5.1. Boundary and initial condition. zG means zeroGradient and wF, wallFunction, mWV movingWallVelocity

Since The purpose of the plenum is only to avoid reflections from the inlet boundary, to improve the accuracy of the results and the stability of the simulation, it has been realized with a coarse mesh in volume and near wall patches as well. Hence it is useless applied wallFunction to Tank patches for any physical quantity. On the other side they

are going to be applied on other wall patches even if they are develop only for completely developed flows and not for ICE or engine like geometry. Internal field for P, T field are set up according to evironmental conditions, while U is set to quiescent velocity with piston position at TDC field, because turbulence is activated by piston motion whose velocity is calculated in z direction with Eq. 5.2. Other physical quantities are set up with values that allows numerical stability in formers time steps. Moreover for all wall patches zeroGradient on Temperature is applied which means that not heat exchange are taken into account and zeroGradient on pressure is applied too.

Particular condition has been applied on inlet: since mass flow rate change in sign due to piston motion, between intake and exhaust stroke, inletOutlet condition has been applied on k and ω to provide a generic outflow condition, with specified inflow for the case of return flow and a similar condition on Temperature as well. Also for U an inlet/outlet boundary condition is applied, working in this manner: A zero-gradient condition is applied for outflow (as defined by the flux); for inflow, the velocity is obtained from the patch-face normal component of the internal-cell value. Notice that this BC on U has been coupled with totalPressure BC on Pressure at inlet makes the problem numerically consistent.

Finite volume interpolation schemes

Interpolation schemes used to discretize governing equation (Sec. 2.1) are:

- a second order backward differencing scheme (Crank Nicholson) has been used to discretize temporal derivative. Even if is unconditionally stable, since stability does not rely on temporal discretization, when a convective term appears in equation, spurious oscillations can appear. Thus a blending with Euler implicit method is performed in order to remove this wiggles.

- pure second order has been used on all space derivatives except for momentum and energy advection, which use a particular scheme, and turbulence , which is usually discretized with one order of accuracy lower than momentum. This allow for a higher accuracy but in many cases, especially with coarse mesh, can be unbounded.

- momentum advection $\nabla \cdot (\rho \mathbf{U}\mathbf{U})$ and energy advection $\nabla \cdot (\rho \mathbf{U}\mathbf{h})$ have been discretized with Linear-upwind stabilized transport (LUST).
  It is a new interpolation schemes in which linear-upwind is blended with linear interpolation to stabilize solutions while maintaining second-order behavior. The blending-factor is set to 0.75 linear which optimizes the balance between accuracy and stability on a range of LES cases with a range of mesh quality. The scheme is proving particularly successful for LES/DES in complex geometries with complex unstructured meshes.

- linear scheme for point to point interpolation;

Algorithm control

Since the compressible dynamic solver is used in Transient-simple mode, Courant number can be higher than 1. However the timestep is limited 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 Co must be lower than 5. Also Topological changes pose some limits on the maximum time step size needed to guarantee topological consistency. If the temporal time step was too large, the mesh handling algorithm may skip the point, when a topological changes should be triggered, leading to a wrong mesh configuration. To avoid this problematic situation 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 $\mathbf{U}_p$ during the compression phase:

$$\Delta z_{piston} = \mathbf{U}_p \cdot \Delta t \tag{5.3}$$

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

$$\Delta z_{piston} = \Delta z_{layer}^{max} \tag{5.4}$$

then the time step is recalculated as:

$$\Delta t_{lim} = \Delta z_{layer}^{max} / \mathbf{U}_p \tag{5.5}$$

Where Δz_{layer}^{max} layer is the layer removal threshold and Δt_{lim} the maximum allowed time step. Thus time step turns out to be limitated by two factors: phase coherence for LES and topological consistency. The first leads to use a small time step with an *adjustableTimeStep* option to avoid that CFL goes beyond its max value, the second is ensure by the aforementioned adaptive topology driven time stepping.

5.1.3 Results

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). 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. Azimuthal and ensembling averaging of the data have been used to calculate statistical quantities to a limited number of engine cycles: because of this, a sufficient number of engine cycles must be simulated to obtain converged statistics. In total, eight engine cycles were calculated and the first was discarded from the statistical analysis, in order to minimize the effect of the initial conditions. This

choice was due to the need of excluding any non-repeatable phenomenon that might occur during the very first phases of the simulation, when the cylinder conditions are very different with respect to the other cycles. Comparisons between simulations and experiments have been carried out at 36° , 90° and 144° CA (intake stroke) for the mean and rms of the axial velocity.

Average and rms velocity

Space averaging has been computed along the azimuthal direction, in order to reduce the total number of engine cycles to be simulated. Data have been sampled with an angular step of 5°; linear interpolation has been used to approximate the fields in between cell centers. Both ensemble averaged velocity $\bar{\mathbf{U}}$ and rms fluctuations (${u_z}^{rms}$) have been extracted from the circumferential-averaged planes and compared with experimental measurements. Profiles have been plotted along the cylinder radius at increasing distance (with a step $\Delta z = 10\,mm$) from the cylinder head (conventionally assumed as $z = 0\,mm$); only the axial component of the velocity has been considered for comparison, since no experimental data were available in [16] for the other two components. Thus In Figures 5.3, 5.4 and 5.5, mean and rms fluctuations of the axial velocity are compared at selected axial distances z from the cylinder head. Both mean and rms fluctuansion are normalised by the average piston speed $\bar{\boldsymbol{U}}_{\boldsymbol{p}}$ defined as $2Sn$.
In Fig. 5.3, velocity profiles at $CA = 36°$ are represented. The amplitude of local minima at $z = -10\,mm$ and $z = -20\,mm$ is correctly estimated, even though the position is slightly shifted towards the liner wall. The same considerations can be done for the local maxima of u_z^{rms} .
At $z = -30\,mm$ flow velocity is very low, because the measurement location is very close to the piston top, thus Both LES and DLRM catch very well experimental data. Comparison of averaged velocities at $CA = 90°$ are represented in Fig. 5.4.
The experimental velocity profile is caught fairly well at $z = -10mm$ and $z = -20mm$, even though the position is slightly shifted towards the liner wall as at $CA = 36°$ at the same height. On the other hand, differences between simulated and measured velocities increase as the sampling plane location is moved far from the cylinder head, even though the qualitative trend is generally preserved. Predictions of velocity field for $CA = 144°$ (Fig. 5.5) show a very good agreement with experimental data, with minor discrepancies near the cylinder walls for rms fluctuations. As shown in [19] at $CA = 90°$ (Fig. 5-b) there is a mismatch.

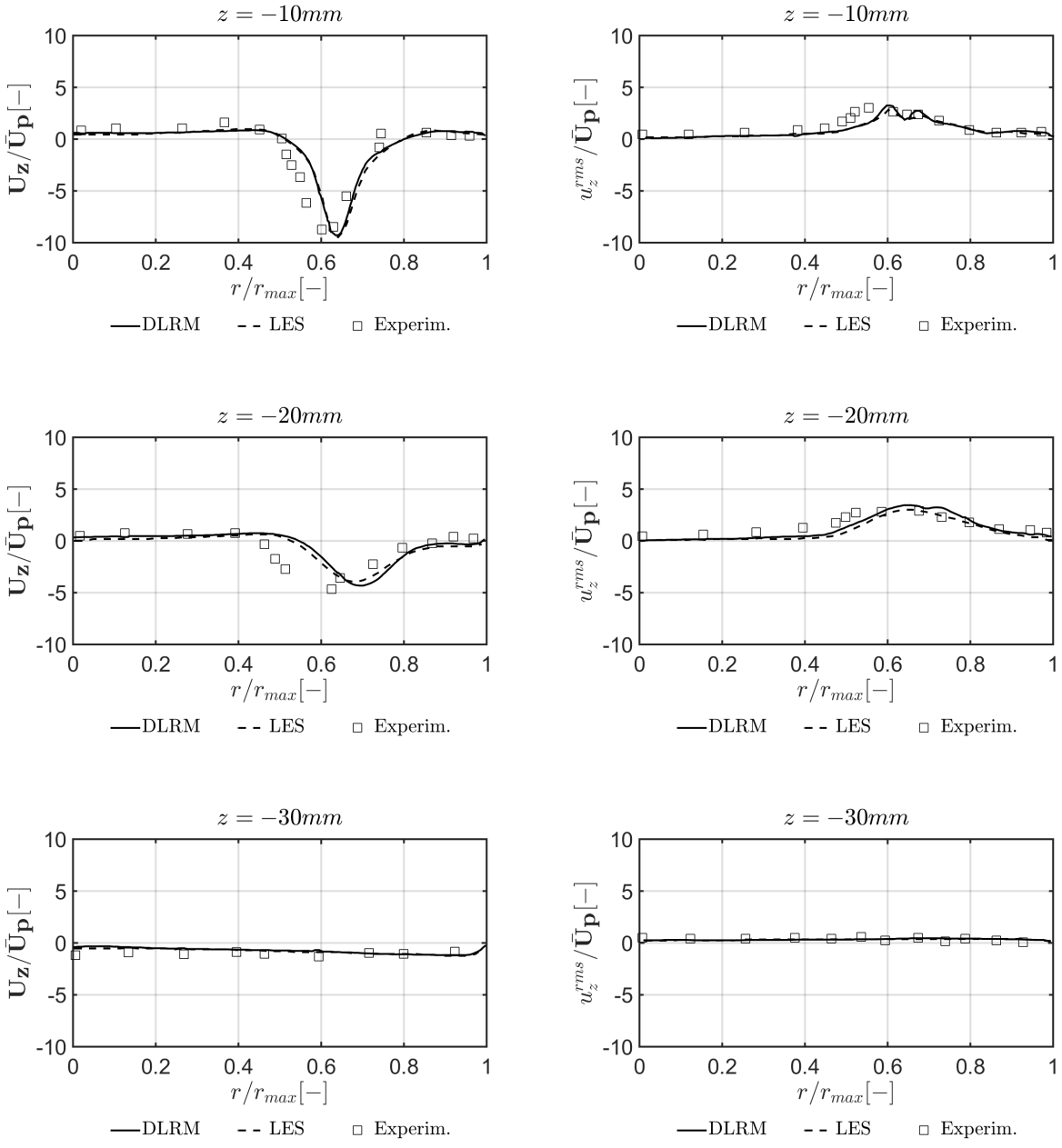


Figure 5.3. Profiles of mean axial velocity (left) and axial RMS fluctuations (right) for $CA = 36°$ ATDC, at different distances from the cylinder head (conventionally $z = 0\,mm$).

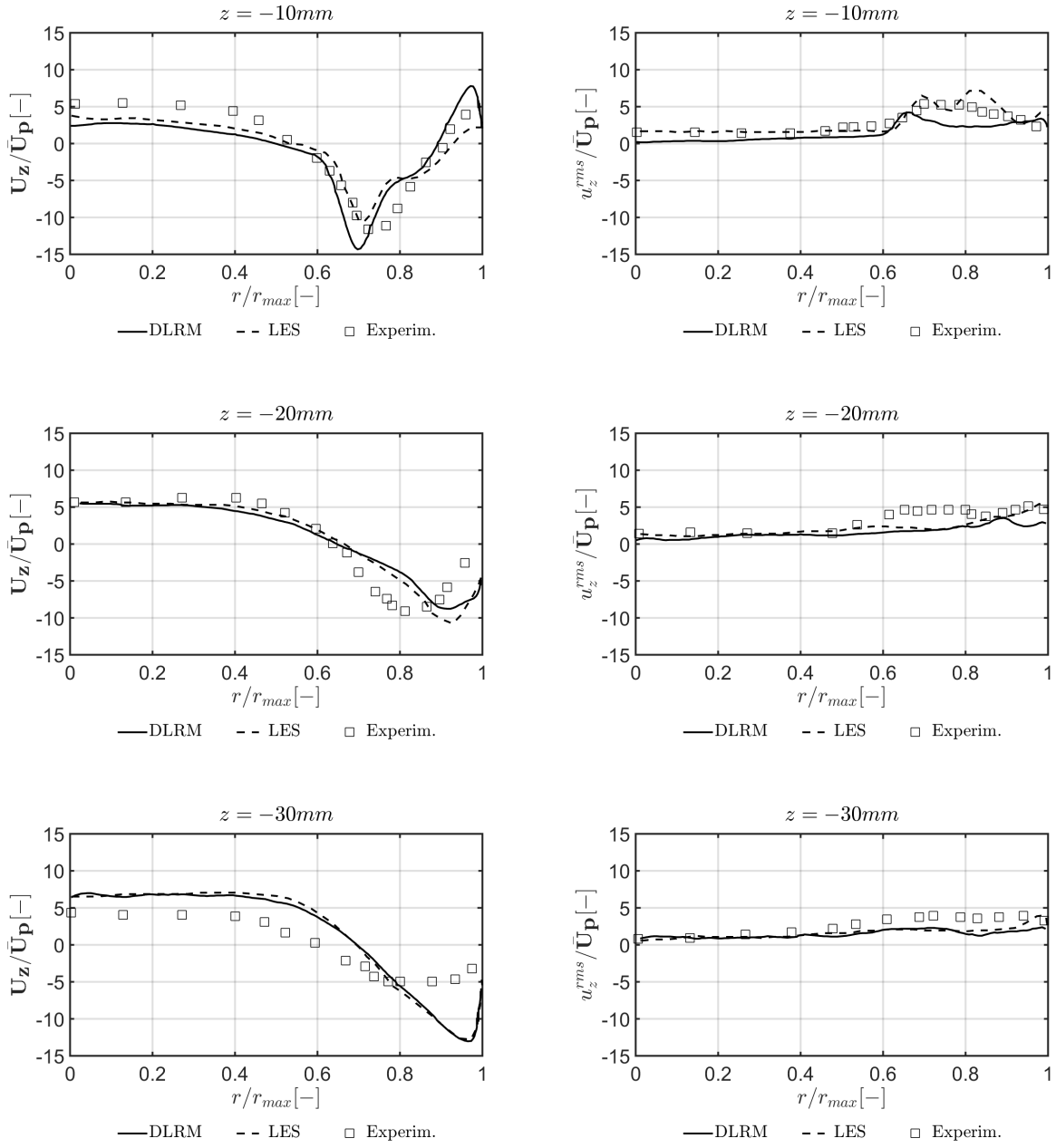


Figure 5.4. Profiles of mean axial velocity (left) and axial RMS fluctuations (right) for $CA = 90°$ ATDC, at different distances from the cylinder head (conventionally $z = 0\,mm$).

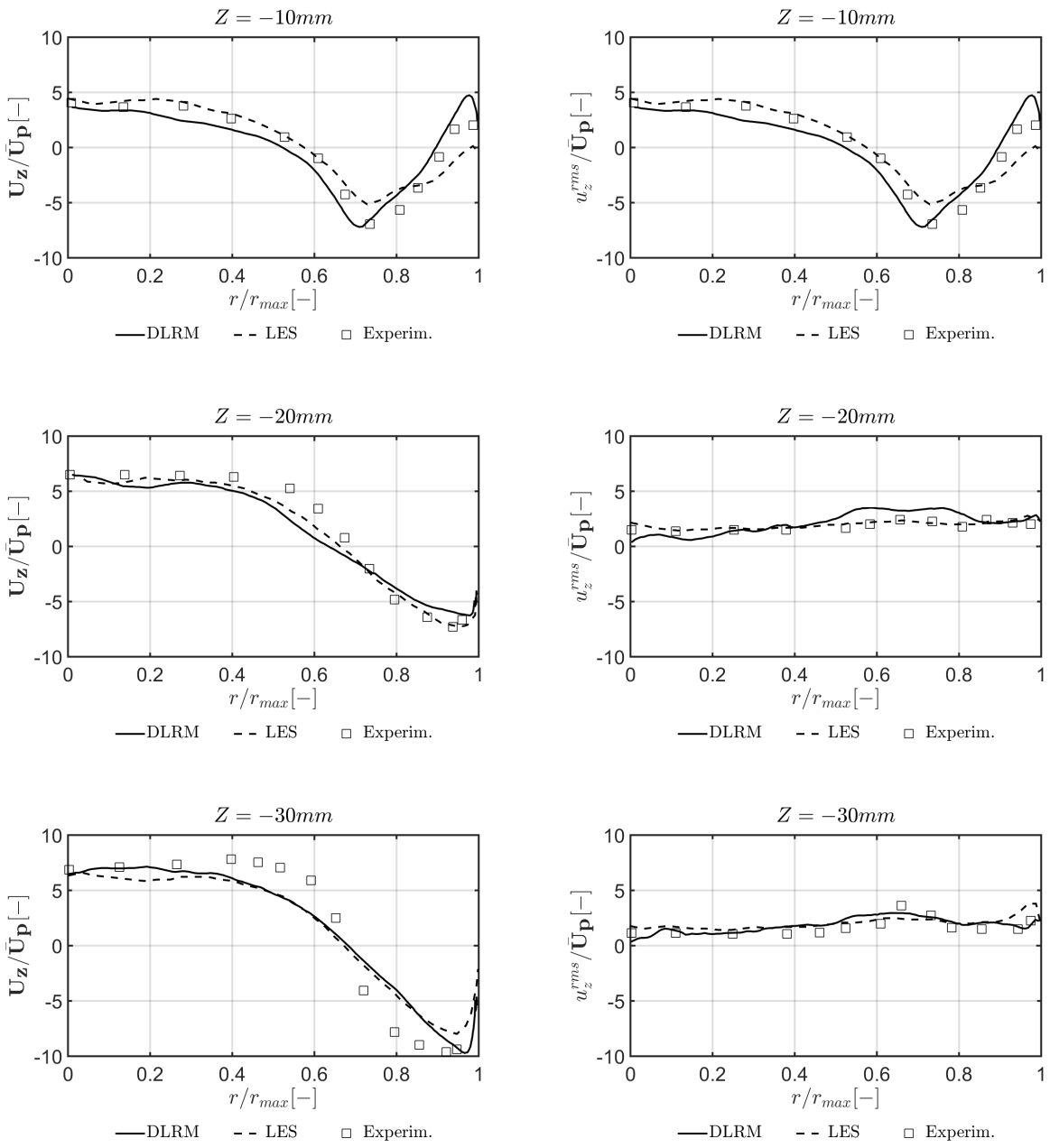


Figure 5.5. Profiles of mean axial velocity (left) and axial RMS fluctuations (right) for $CA = 144°$ ATDC, at different distances from the cylinder head (conventionally $z = 0\,mm$).

A possible explanation may be found in the mesh resolution that was used at this CA which is not sufficient to capture the main vortex structures influencing the flow

field. Since the mesh used for DLRM has the 25 % of resolution of that used for LES, if this phenomena was presented with LES , using DLRM has been accentuated. DLRM values of $\mathbf{U_z}$ further from piston wall, at 90° ATDCE, show a little mismatch from LES values. It rely on higher g^2 value owned at that CA , whose possible explanation are:

- lower resolution of mesh than a LES one;
- bad temporal correlation between following time steps;
- highest velocity flow field at 90° that together with coarse mesh, makes hard to catch the velocity gradient in the shear layer between the annular jet and the in-cylinder region and on the liner walls where the jets impacts.

A similar behavior can be found at 144° ATDCE since jet velocity is still high enough. Hence, similarly to what has been found in [19], 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.

5.2 TCC case

5.2.1 Experimental set-up

The Transparent Combustion Chamber (TCC), whose history has been shown in Sec.4.3.2 engine is an optical engine that was set up at the University of Michigan by [20] 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. 5.2

Bore	92 mm
Stroke	86 mm
Connecting rod length	234.95 mm
TDC clearance height	9.5
Geometric compression ratio	10
Engine speed	1300 RPM

Table 5.2. geometrical feature and relevant data for TCC-III

The geometry has been discretized with a pure hexa mesh as the previous case, and was presented in Sec. 4.3.2. At the time the thesis was written only a compression test on cylinder of Fig. 5.6 has been performed: results has been compared with the experimental.

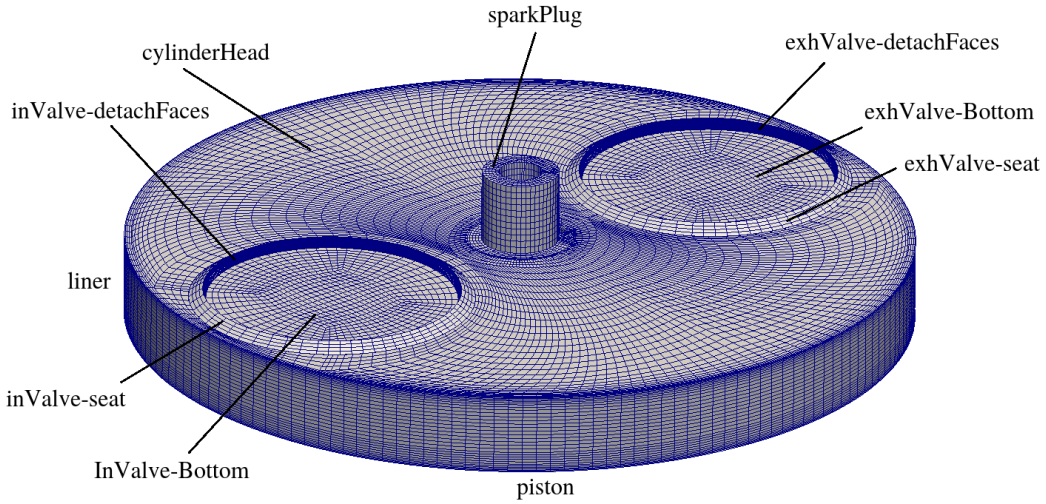


Figure 5.6. Mesh of detach cylinder for compression test

Experimental cylinder pressure data and T cylinder data are provided for 235 motored cycles (four strokes). Pressure has been sampled, using a highly-sensitive piezoelectric combustion pressure sensors , every 0.5° of CA. Hence to compare CFD results with experimental data a mean pressure over entire volume has been computed. Then, This mean pressure , is compared with the mean P-cylinder over 235 cycle for each CA. Only about 200 CAD has been simulated. Firstly the mesh in Fig.5.6 is moved, without fluid dynamic, to $\theta = 100\,CAD$ where the real compression start. Thus $T(\theta)$ and $P(\theta)$ inside the cylinder at this CAD are use as initial condition, where $P(\theta)$ and $T(\theta)$ are the mean P and T over 235 cycles at $\theta = 100\,CAD$

Case setup

The boundary and initial condition for involved physical quantities are presented briefly in Tab. 5.3

	sparkPlug	liner	cylinderHead, /dtachFaces, /valve/	/piston	internalField
k	zG	wF	wF	wF	$0.01\ m^2/s^2$
ω	zG	wF	wF	wF	$400\ s^{-1}$
μ_t	zG	wF	wF	wF	$1e-8\ kg/ms$
α_t	zG	wF	wF	wF	$0\ kg/ms$
P	zG	zG	zG	zG	$154330\ Pa$
T	zG	Tcyl_surf	zG	zG	$369.83\ K$
U	$0\ m/s$	$0\ m/s$	mWV	$0\ m/s$	$0\ m/s$

Table 5.3. Boundary and initial condition at 100 CAD. zG means zeroGradient and wF, wallFunction, mWV movingWallVelocity

Numerical schemes

The same numerical schemes of previous case has been adopted with some modifications on derivatives where the "limited corrected" option is applied to laplacian and to the orthogonal part of gradient to contain the "false diffusion" terms arises from non-orthogonality due to complex unstructured mesh generated for TCC case.

Algorithm control

The same set-up of previous case has been adopted, but, since only flow field inside cylinder is computed, there are not the presence of very small cells near valve with high velocity which would make the local CFL really high. Thus the conditon of phase coherent for LES is easily respected, while the topology consistency is always guarantee by adaptive topology driven time stepping.

5.2.2 Results

The curve of compression $p(\theta)$ obtained with p probed is compared with experimental data and with an adiabatic compression from intake valve closure (IVC) to Exhaust Valve Opening (EVO). As expected the cylinder pressure trace (Fig. 5.7) has been correctly predicted, both during the compression and the expansion phase.
Also the instantaneous volume is trace to verify that dynamic addition and removal of cell layers was consistent; 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 during expansion, as the cell thickness was higher than 1 mm.

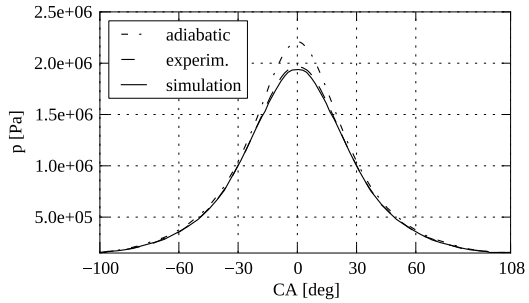


Figure 5.7. Average in-cylinder pressure of the motored engine from IVC (≈ 100° BTDC) to EVO (≈ 108° ATDC)

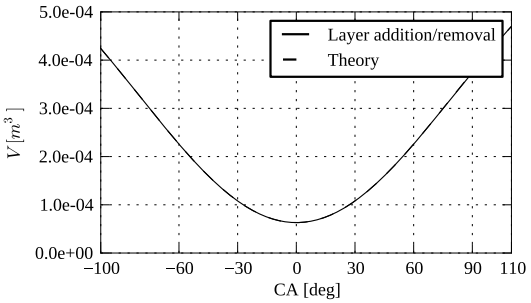


Figure 5.8. experimental in-cylinder volume during the com- pression and expansion phase with closed valves (calculated vs predicted)

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

$$V = V_c + s_p(\theta) \cdot A_c \tag{5.6}$$

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 calculated as:

$$s_p(\theta) = \frac{S}{2}[1 + 1/\Lambda - cos\theta - 1/\Lambda\sqrt{(1 - \Lambda^2 sin^2\theta)}] \tag{5.7}$$

$$\Lambda = R/L \tag{5.8}$$

where S=stroke, R=crank radius and L=connecting rod length. 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). 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. 5.8, 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.

Chapter 6

Conclusions

The aim of this thesis work was to define a methodology in the commercial software ANSYS ICEM CFD to generate high-quality oriented multi-block hexahedral unstructured grids based on non-conformal interfaces to perform dynamic simulations in OpenFOAM.

At the time the thesis is written, no information about generation of this kind of grids in ICEM was available. In particular, in the meshing approach proposed in this work:

- the multi-block hexahedral domain is generated by a top-down approach;
- oriented mesh zones are generated along internal baffles (generated in ICEM) that are removed by ad-hoc applications in OpenFOAM;
- the mesh generation strategy is fully compatible with the extensions of the dynamic mesh library of OpenFOAM, released by OpenFOAM Foundation.

The proposed methodology has been applied to generate two different engine geometries. Simulations of turbulent flows performed on the two grids by the DLRM model were able to capture the main features of turbulent in-cylinder flows and to be accurate to predict the fluid dynamic quantities even with a coarse mesh. Results show that DLRM is able to provide very reliable predictions about the in-cylinder flow with reasonably coarse grids if compared to LES. Also compression test on TCC shows good results both for cylinder pressure trace and the instantaneous volume which mean very good performances of solver used.

Finally, the methodology proposed is absolutely general and it can be applied also to non-engine geometries. For this reason, future works will be focused on the simulation of engines as well as on the simulation of fuel injectors.

References

[1] H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics The Finite Volume Method*. 2007. ISBN ISBN: 9780131274983.

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

[3] D.C. Haworth and K. Jansen. Large-eddy simulation on unstructured deforming meshes: towards reciprocating ic engines. *Computers and Fluids*, 29(5):493 – 524, 2000. ISSN 0045-7930. doi: http://dx.doi.org/10.1016/S0045-7930(99)00015-8. URL `http://www.sciencedirect.com/science/article/pii/S0045793099000158`.

[4] Kai Liu and D.C. Haworth. Large-eddy simulation for an axisymmetric piston-cylinder assembly with and without swirl. *Flow, Turbulence and Combustion*, 85 (3-4):279–307, 2010. URL `http://dx.doi.org/10.1007/s10494-010-9292-1`.

[5] B. Enaux, V. Granet, O. Vermorel, C. Lacour, C. Pera, C. Angelberger, and T. Poinsot. Les study of cycle-to-cycle variations in a spark ignition engine. *Proceedings of the Combustion Institute*, 33(2):3115 – 3122, 2011. ISSN 1540-7489.

[6] Christian Hasse, Volker Sohm, and Bodo Durst. Numerical investigation of cyclic variations in gasoline engines using a hybrid urans/les modeling approach. *Computers and Fluids*, 39(1):25 – 48, 2010. ISSN 0045-7930.

[7] L. Thobois, G. Rymer, T. Soul'eres, T. Poinsot, and Van den Heuvel B. Large-eddy simulation for the prediction of aerodynamics in ic engines. *Internation journa Vehicle Design*, 39(4):368–382, 2005. doi: 10.1504/IJVD.2005.008468.

[8] F. Piscaglia, A.Montorfano, and A. Onorati. A scale adaptive filtering technique for turbulence modeling of unsteady flows in ic engines. *SAE Technical Paper n. 2015-01-0395*, 2015. doi:10.4271/2015-01-0395.

[9] F. Piscaglia, A.Montorfano, and A. Onorati. A moving mesh strategy to perform adaptive large eddy simulation of ic engines in openfoam. 2014.

[10] W.W. Gyllenram and H.H. Nilsson. Design and validation of a scale-adaptive filtering technique for lrn turbulence modeling of unsteady flow. *ASME. Journal of Fluids Engineering*, 130, 2008. doi: http://dx.doi.org/10.1115/1.2911685.

[11] F. Brusiani and G.M. Bianchi. Basic numerical assessments to perform a quasi-complete les toward ic-engine applications. *ASME. Fluid Flow, Heat Transfer and Thermal Systems*, 7, 2010.

[12] 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 n. 2015-01-0384*, 2015. doi:10.4271/2015-01-0384.

[13] J. H. Ferziger and M. Peri c. *Computational Methods for Fluid Dynamics*.

[14] P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, 1979. doi: 10.2514/3.61273.

[15] 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 n. 2013-24-0027*, 2013. doi:10.4271/2013-24-0027.

[16] A. P. Morse, J.H. Whitelaw, and M.M. Yianneskis. Turbulent flow measurements by laser-doppler anemometry in motored piston-cylinder assemblies. *Journal of Fluids Engineering*, 101(2):208–216, 1979.

[17] D. C. Haworth. Large-eddy simulation of in-cylinder flows. *Oil & Gas Science and Technology - Rev. IFP*, 54(2):175–185, 1999. doi: 10.2516/ogst:1999012. URL `http://dx.doi.org/10.2516/ogst:1999012`.

[18] V. Mittala, S. Kang, E. Doran, D. Cook, and H. Pitsch. Large-eddy simulation for an axisymmetric piston-cylinder assembly with and without swirl. *Oil, Gas Sci.Techn IFP Energies nouvelles International Conference: LES4ICE 2012*, 69 (2):29–40, 2014.

[19] A. F. Piscaglia, Montorfano 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 n. 2014-01-1147*, 2014. doi:10.4271/2014-01-1147.

[20] Engine Combustion Network. `http://www.sandia.gov/ecn/engines/engineFlows/TCCEngine.php`. 2013.

[21] D. Reuss, T. Kuo, B. Khalighi, and D. Haworth. Particle image velocimetry measurements in a high-swirl engine used for evaluation of computational fluid dynamics calculations. *SAE Technical Paper n. 952381*, 1995. doi:10.4271/952381.

[22] D. Reuss. Cyclic variability of large-scale turbulent structures in directed and undirected ic engine flows. *SAE Technical Paper 2000-01-0246*, 2000. doi:10.4271/2000-01-0246.

[23] Tang-Wei Kuo and David L. Reuss. Multidimensional port-and-cylinder flow calculations for the transparent-combustion-chamber engine. volume 23, pages 19–29, 1995. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0029223651&partnerID=40&md5=96384c87df9e4b9ade49fa2976f94984`. cited By 7.

[24] D.L. Reuss and M. Rosalik. Piv measurements during combustion in a reciprocating internal combustion engine. In R.J. Adrian, D.F.G. Durao, F. Durst, M.V. Heitor, M. Maeda, and J.H. Whitelaw, editors, *Laser Techniques Applied to Fluid Mechanics*, pages 441–456. Springer Berlin Heidelberg, 2000. ISBN 978-3-642-63087-3. doi: 10.1007/978-3-642-56963-0_29. URL `http://dx.doi.org/10.1007/978-3-642-56963-0_29`.

[25] Kuo, Tang-Wei, Yang, Xiaofeng, Gopalakrishnan, Venkatesh, and Chen, Zhaohui. Large eddy simulation (les) for ic engine flows. *Oil Gas Sci. Technol. – Rev. IFP Energies nouvelles*, 69(1):61–81, 2014. doi: 10.2516/ogst/2013127. URL `http://dx.doi.org/10.2516/ogst/2013127`.

[26] V. Sick. A common engine platform for engine les development and validation. *LES4ICE*, 2010.

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

[28] Ansys icem cfd tutorial's manual, .

[29] Ansys icem cfd user's manual, .

[30] The openfoam foundation. `www.openfoam.org`, . 2015.

[31] Openfoam foundation - openfoam user's guide, v2.3.0. `www.openfoam.org`, . 2015.

[32] G. Ferrari. *Motori a combustione interna.*