POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale
Dipartimento di Energia

Corso di Laurea Magistrale in Ingegneria Energetica

# MULTI-DIMENSIONAL STEADY STATE IN-CYLINDER FLOW SIMULATION BY AN OPEN-SOURCE CFD CODE

| | |
|---|---|
| Relatore: | Prof. Federico PISCAGLIA |
| Correlatore: | Ing. Andrea MONTORFANO |

Tesi di Laurea di:
Jacopo ZISA Matr. 770978

Anno Accademico 2012-2013

*You say the hill's too steep to climb, climb it!*
"Fearless" - Pink Floyd

# Ringraziamenti

Desidero ringraziare il professor Federico Piscaglia e l'ingegner Andrea Montorfano per le conoscenze che hanno saputo trasmettermi, per la disponibilità e la pazienza dimostrata ogni volta che mi hanno visto entrare nel loro ufficio, seguendomi in ogni passo di questo lavoro con preziosi consigli.

Ringrazio i miei genitori e mio fratello che mi hanno sempre sostenuto in ogni occasione, permettendomi di completare il corso di studi e raggiungere un obbiettivo importante come questo.

Un ringraziamento particolare va alla mia ragazza, per l'aiuto immenso che mi ha dato e se questo pezzo di carta è scritto, buona parte del merito è suo.

Infine, *last but not least*, ringrazio i miei compagni tesisti Mattia e Tommaso, che hanno condiviso con me questo percorso e mi hanno permesso di sdrammatizzare ogni momento.

# Contents

# List of Figures

# List of Tables

# Abstract

In this work, a series of steady state flow simulations, with the CFD software OpenFoam, is carried out for several valve lifts in order to describe air movement characteristics in cylinder and intake port of a spark ignition engine. In-cylinder flow is studied by means of steady state RANS turbulence models, using the finite volume method. Investigation is aimed at analyzing the generation and the evolution of flow field during the intake stroke and checking the ability to realize arranged motions in the cylinder. The setting of numerical simulation model is consistent with boundary conditions of steady state flow experimental tests in order to compare the results, represented by discharge coefficient and tumble coefficient. Simulation results show that tumble is the main in-cylinder flow motion and tumble vortex grows in size and intensity by the increasing air entering through the valves. Moreover, discharge coefficient and tumble coefficient can be compared with experimental tests with good matching.

**Keywords:** *CFD, OpenFOAM, discharge coefficient, tumble coefficient, steady state flow*

# Sommario

I miglioramenti nella potenza di calcolo hanno portato a rapidi progressi nella simulazione fluidodinamica computazionale (CFD) dei motori a combustione interna (MCI). L'utilizzo di simulazioni fluidodinamiche consente una maggiore comprensione della dinamica del flusso all'interno dei MCI. Ogni nuovo progetto sviluppato è il frutto di un lavoro parallelo tra la parte di simulazione e le prove effettuate attraverso un banco di flusso. Uno dei principali vantaggi della CFD rispetto ai test sperimentali consiste nel fatto che non è necessario costruire modelli in scala di solito estremamente costosi e che presentano spesso condizioni di funzionamento difficili da replicare. La CFD permette inoltre di ridurre la quantità di prove sperimentali portando ad una riduzione dei tempi e quindi dei costi. L'obiettivo di questa simulazione è proprio quello di riprodurre una prova di flussaggio stazionario ricavando i parametri più significativi per la motoristica, ovvero il coefficiente d'efflusso della valvola e l'intensità del vortice di tumble che si genera nel cilindro. Per poter analizzare in modo completo la fase di aspirazione e per tracciare i grafici del coefficiente d'efflusso e di intensità del vortice di tumble, la geometria di cui si dispone è rappresentata da un modello con differenti valori di alzata della valvola; nello specifico le alzate di cui si dispone vanno da 1 mm a 9 mm. In tal modo, impostando il flussaggio su ogni alzata, è possibile ricostruire il comportamento del fluido durante tutta l'aspirazione. Una volta ultimate tutte le simulazioni i risultati acquisiti verranno confrontati con quelli raccolti dalle indagini sperimentali.

Per questo lavoro di tesi si è utilizzato *OpenFOAM*, un software con licenza GNU *GeneralPublicLicense*, per il quale è possibile sviluppare autonomamente applicazioni e librerie personali. Le librerie sono scritte in linguaggio C++ e possono essere riferite a due principali categorie: i *solutori* che hanno implementato al loro interno le equazioni necessarie a risolvere specifici problemi e le *utility*, applicazioni accessorie necessarie per la manipolazione dei dati sia per quanto riguarda il $pre-processing$ sia per il $post-processing$.

Una delle prime operazioni da eseguire per procedere con l'analisi fluidodinamica è la preparazione della geometria, che consiste nella creazione della griglia di calcolo tramite l'identificazione di volumi di controllo sul dominio. La griglia può essere più o meno raffinata, presenta cioè zone più fitte rispetto ad altre, a seconda dell'accuratezza richiesta. In generale si cerca di ottenere una griglia più fitta nelle zone dove sono presenti i maggiori gradienti, in modo da avere una rappresentazione il più possibile vicina al fenomeno reale. Tuttavia è importante ottimizzare il dominio di calcolo in modo da avere soluzioni in tempi accettabili. Data la complessità della geometria della testa cilindro fornita attraverso file CAD sono state necessarie molte prove per l'ottenimento di un risultato soddisfacente per la simulazione CFD. La geometria è data da un cilindro completo di valvole di aspirazione e condotto. Si tratta quindi dello stesso assemblato di cui si dispone nel caso di una prova sperimentale. Prima di tutto è stata creata una background mesh di celle esaedriche con l'utility *blockMesh*, definendo un grado di raffinamento generale, viene lasciato agli step successivi l'eventuale raffinamento locale. Successivamente si è passati ad utilizzare l'utility *snappyHexMesh* per ottenere una mesh conforme ai tipici valori di qualità richiesti da una simulazione CFD su una geometria complessa. Questa utility viene controllata attraverso il dizionario *snappyHexMeshDict* associato, a cui vengono forniti i parametri necessari per ogni step di generazione della griglia di calcolo e viene eseguito un ciclo qui descritto. Il primo step riguarda la generazione di una *castellatedMesh*, che rappresenta una griglia di calcolo con l'introduzione del raffinamento locale ma senza *morphing* della testa cilindro, eliminando le celle esterne alla geometria in analisi. Il secondo step, detto *snap*, procede con la generazione della mesh con morphing della geometria che consiste nello spostamento dei vertici delle celle sul confine della castellatedMesh sulla superficie del file STL. In questa fase viene successivamente rilassata la mesh interna basandosi sull'ultimo spostamento dei vertici delle celle di confine. Vengono inoltre identificati i vertici che causano una violazione dei parametri di qualità imposti nella sezione *MeshQualityControl*. Infine viene ridotto lo spostamento di questi vertici dal loro valore iniziale e si ripete il ciclo fino al raggiungimento dei paramenti di qualità richiesti. L'ultimo step riguarda il *layerAddition*. In quest'ultima fase viene aggiunto un layer di celle sulle patch di contorno. Si tratta di un processo opzionale che introduce ulteriori celle esaedriche allineate con le superfici. L'inserimento del boundary layer comporta il restringimento della mesh al confine con la superficie di contorno per permettere l'inserimento di queste celle. La mesh generata consiste in una griglia di calcolo non strutturata, costituita prevalentemente da esaedri regolari, mentre le restanti celle risultano essere dei tetraedri. I principali parametri da controllore nello sviluppo di una mesh sono la skewness, la non-ortogonalità e aspect ratio. Il primo parametro misura il grado di deformazione della cella. Il secondo descrive la giacitura normale o inclinata alla faccia del segmento che congiunge due centri cella adiacenti. L'ultimo indica il rapporto tra lunghezza massima e minima dei lati di una cella.

Per risolvere un sistema di equazioni differenziali alle derivate parziali è necessario assegnare le condizioni al contorno in modo opportuno. Tutte le simulazioni sono state effettuate in due fasi. Nel primo step viene fissato il salto di pressione pari a 1.1 come condizione al contorno, in modo da ricavare la portata circolante. In questo caso, una condizione di pressione totale è stata fissata nella patch di inlet e una condizione di pressione statica nella patch di outlet (BC1). Nel secondo step la portata massica appena calcolata viene applicata come condizione al contorno di inlet, mantenendo quella già assegnata precedentemente come condizione al contorno di outlet (BC2). L'utilizzo del secondo setting di condizioni al contorno permette di raggiungere una migliore convergenza sui residui e generalmente una migliore stabilità alle prime iterazioni. Le grandezze del modello turbolento, quindi $k$ ed $\omega$ sono state inizializzate con riferimento alla letteratura. Questa inizializzazione tuttavia, specialmente per le basse alzate delle valvole (1 mm, 2 mm, 3 mm), ha portato ad un arresto della simulazione nelle prime iterazioni, mentre per le alzate medie e grandi (da 4 mm a 9 mm), la simulazione ha raggiunto la convergenza. Successivamente a questa prima fase di prove, come spiegato in precedenza, le nuove condizioni al contorno BC2 sono state aggiornate con i nuovi valori di portata, $k$ ed $\omega$. Questi nuovi valori delle grandezze turbolente sono stati ricavati mediando i valori delle celle sulla patch di outlet dopo che la simulazione ha raggiunto la convergenza. Quindi, al fine di inizializzare nel modo corretto anche $k$ ed $\omega$ alle basse alzate delle valvole, è stata eseguita una estrapolazione lineare con il valore di portata massica interpolato. Conoscendo i valori esatti sia di $k$ ed $\omega$ sia della portata alle alzate 4 mm e 5 mm e il valore interpolato di portata all'alzata 3 mm, attraverso una estrapolazione lineare sono stati ricavati i valori di $k$ ed $\omega$ per l'alzata 3 mm della valvola e sono stati assegnati alle condizioni al contorno BC1.

Ultimate tutte le simulazioni, grazie a questo lavoro è stato possibile descrivere il comportamento del flusso d'aria nella sezione di aspirazione di un motore ad accensione comandata per diverse alzate delle valvole. I risultati ottenuti possono essere giudicati affidabili, confrontati con le prove sperimentali, sopratutto per le alzate intermedie (da 3 mm a 7 mm) con un buon confronto delle mappe di velocità all'interno del cilindro. A conferma del buon lavoro svolto, anche l'andamento del coefficiente di efflusso e del numero di tumble è comparabile in maniera più che discreta con le prove sperimentali in questo range di alzate. Le differenze riscontrate tra i risultati numerici e quelli sperimentali sono da attribuire a diversi fattori. Le simulazioni di CFD sono approssimazioni del fenomeno fisico per cui anche i risultati saranno approssimati e limitati dalle risorse disponibili. A causa delle moderate risorse hardware a disposizione e della complessità della geometria analizzata, è stato necessario trovare un compromesso tra raffinamento della mesh e tempo di calcolo. Con modelli di turbolenza di tipo $k$ - $\epsilon$ o suoi modelli derivati, griglie più lasche possono essere utilizzate, tuttavia per garantire una $y^+ > 30$ le dimensioni delle celle non risultavano più compatibili con il raffinamento richiesto dalla sezione ristretta della valvola. Inoltre questo modello di turbolenza descrive in maniera poco completa flussi caratterizzati da elevati gradienti di pressione e ricircoli. Il modello di turbolenza $k$ - $\omega$ si è dimostrato adatto e flessibile per questo tipo di lavoro. Parlando della generazione della mesh, l'uso di *snappyHexMesh* ha portato ad avere alcuni problemi legati al fatto che è una utility basata su un processo di modellazione della mesh iterativo e automatico. Il problema principale è dovuto al fatto che la deviazione angolare della direzione del flusso dal vettore che congiunge i due centri cella può generare diffusione numerica. Un progetto di mesh con celle orientate nella stessa direzione del flusso, nella zona ristretta della valvola, potrebbe rappresentare una soluzione a questo problema. Tuttavia, aggiungere il boundary layer senza prima aumentare il raffinamento a parete, ha generato delle celle distorte con valori di non-ortogonalità e skewness molto alti. Procedendo invece con un ulteriore raffinamento a parete, la mesh non presentava più celle distorte ma un numero di elementi eccessivo per i mezzi di calcolo a disposizione. Inoltre, le differenze osservate comparando i risultati possono anche essere causate da un leggera differenza della geometria CAD, rispetto alla flow box testata sul banco fluidodinamico, in quanto si sono rese necessarie delle modifiche per adattare la geometria alle prove di simulazione.

Per tutte queste ragioni, è possibile affermare che la progettazione di un particolare dispositivo preveda uno percorso condotto parallelamente tra le simulazioni numeriche e le indagini sperimentali, in modo da ottenere un studio più completo del fenomeno investigato.

# Chapter 1

# Introduction

Improvements in computer processing power and fluid simulation codes have resulted in rapid advancements in computer-based engine simulation. The use of three-dimensional computational fluid dynamic (CFD) codes allows for greater understanding of flow dynamics in internal combustion engine system, before any prototype is manufactured. In fact, many manufacturers are routinely using CFD as part of their engine design process. Steady state CFD simulation comparison with steady state flow bench results has been widely applied in academic and industrial research. A key advantage of numerical simulation compared to experimental tests is that in the first case it is not necessary to build scale models, usually extremely expensive, or fulfill analysis of systems in difficult conditions to replicate. This reduces the amount of experimental tests necessary to develop a new product, which ultimately reduces cost and time.

In this work, a series of steady state flow cold simulations is carried out for nine valve lifts (1 mm to 9 mm), in order to describe air movement characteristics in cylinder and intake port of a spark ignition engine. Investigation is aimed at analyzing the generation and the evolution of flow field during the intake stroke at different valve lifts and checking the ability to realize arranged motions in the cylinder. The setting of numerical simulation model is consistent with boundary conditions of steady state flow experimental tests in order to compare the results, represented by discharge coefficient and tumble coefficient.

The intake mass flow rate by an internal combustion engine (ICE) is one of the most important factors for an engine project. Valves and ducts play an important role in the design of internal combustion engines, and influence the performance in the period they are open. The steady state flow data are representative of the dynamic flow behavior of the valve in an operating engine. It has been shown that over the normal speed range, steady flow discharge coefficient results can be used to predict dynamics performance with reasonable precision. The intake system is efficient when there is a minimal difference between the geometrical passage area and the effective flow area.

A practical approach to improve the engine stability is to shorten the combustion duration. This can be achieved by enhancing the tumble motion within the engine cylinder, which enhances the flow turbulence. Generating a significant vortex flow inside the ICE cylinder during the intake process produces high turbulence intensity during the later stages of compression stroke leading to burning rate. With steady state simulations it is possible to describe the variation in size and intensity of tumble vortex for each valve lift.

*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 thesis is organized according to the following pattern. Firstly the physics of the problem and the numerical approach to model it are presented. Than a brief introduction reports the experimental investigation. Afterwards simulation setup is discussed, in order to explain boundary condition, numerical schemes, numerical solution and algorithm control. Mesh generation chapter illustrates grid generation and mesh quality metrics parameters. Subsequently the numerical results are shown in several images that describe the motion features and numerical and experimental results are compared. Finally, in the conclusions section an evaluation of the work and future developments are presented.

# Chapter 2

# Governing Equations and The Finite Volume Method

## 2.1 Governing Equations

The Fluid behavior 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 *(rho, $\mu$, 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 Leibnitz), 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. For steady state problem like in this case, the contribution of time variation is null. In contrast, 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.

It is possible to write the conservation equation of the generic physical property $\varphi$, defining $\Omega$ as control volume delimited by boundary surface S, $\mathbf{n}$ as the surface normal vector of S, $\mathbf{u}$ as the fluid velocity and $Q_\varphi$ as the generic source of $\varphi$.

$$\frac{d}{dt}\int_\Omega \rho\varphi(\mathbf{x},t)\,d\Omega = \int_\Omega Q_\varphi\;d\Omega \tag{2.1}$$

the first member of the equation (2.1) represents the total variation in time of $\varphi$:

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

Then equation (2.1) can be rewritten as:

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

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

$$\int_S \rho\varphi\mathbf{u}\cdot\mathbf{n}dS = \int_\Omega \nabla\cdot(\rho\varphi\mathbf{u})d\Omega \tag{2.4}$$

Finally, if control volume is constant with time, it is possible to express the balance equation for $\varphi$ in the indefinite form, that is expressed by the partial derivatives without reference to a particular volume:

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

### 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:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\varphi\mathbf{u}) = 0 \tag{2.6}$$

If the fluid is incompressible ($\rho$ = constant), the equation can be simplified as:

$$\nabla\cdot\mathbf{u} = 0 \tag{2.7}$$

### 2.1.2 Momentum equation

Momentum equation of a compressible fluid can be written as:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\mathbf{u}d\Omega+\int_{S}\rho\mathbf{u}\mathbf{u}\cdot\mathbf{n}dS=\int_{S}\sigma\cdot\mathbf{n}dS+\int_{\Omega}\rho\mathbf{f}d\Omega \tag{2.8}$$

this equation can be rewritten with Gauss theorem as:

$$\frac{\partial\rho\mathbf{u}}{\partial t}+\nabla\cdot(\rho\mathbf{u}\mathbf{u})=\nabla\cdot\boldsymbol{\sigma}+\rho\mathbf{f} \tag{2.9}$$

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

$$\sigma_{ij}=(-p+2\lambda\nabla\cdot\mathbf{u})\delta_{ij}+\tau_{ij} \tag{2.10}$$

where $\delta_{ij}$ is Kronecker's delta. $\tau_{ij}$ is the viscous stress tensor, which depends on the fluid type. Introducing Newtonian fluid hypothesis, stress tensor can be rewritten as:

$$\tau_{ij}=2\mu S_{ij}+\lambda S_{kk}\delta_{ij} \tag{2.11}$$

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

$$S_{ij}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{2.12}$$

the hydrostatic part of stress tensor can be written as:

$$\frac{1}{3}\sigma_{kk}=-p+\lambda S_{kk}+\frac{2}{3}\mu S_{kk} \tag{2.13}$$

For incompressible fluids, since continuity equation (2.7), the hydrostatic part of $\sigma_{ij}$ is identically equal to the pressure p.

For compressible fluids $\nabla\cdot\mathbf{u}\neq 0$ and since Stokes hypothesis, it is assumed that:

$$\lambda+\frac{2}{3}\mu=0 \tag{2.14}$$

so that the viscosity enters only the deviatoric part of $\sigma$ and the hydrostatic part is equal to the thermodynamic pressure everywhere. With these considerations, the momentum equations can be written as:

$$\frac{\partial\rho\mathbf{u}}{\partial t}+\nabla\cdot(\rho\mathbf{u}\mathbf{u})=\nabla p+\nabla\cdot(\mu\mathbf{S})+\rho\mathbf{f} \tag{2.15}$$

### 2.1.3 Energy equation

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

$$h = u + \frac{p}{\rho} \tag{2.16}$$

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

$$q = -k\nabla T \tag{2.17}$$

where $k$ is thermal diffusivity:

$$k = \frac{\mu c_P}{Pr} \tag{2.18}$$

and $\mu$ is dynamic viscosity, $c_P$ is specific heat at constant pressure and Pr is Prandtl number. Finally, energy equation can be written as:

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) = \frac{\partial p}{\partial t} + \nabla(k\nabla T) + \mathbf{u} \cdot \nabla p + \sigma : \nabla \mathbf{u} \tag{2.19}$$

Where $\partial$p/$\partial$t+$\mathbf{u}$·$\nabla$p is the work made by pressure forces and $\sigma$:$\nabla\mathbf{u}$ is the work made by viscous stresses.

With the further hypothesis of constant specific heat, for incompressible fluid, the energy equation can be rewritten in temperature terms:

$$\frac{\partial \rho T}{\partial t} + \nabla \cdot (\rho \mathbf{u} T) = \nabla \cdot (\mu Pr \nabla T) \tag{2.20}$$

### 2.1.4 State equation

The air state equation must be added to the first three equations: since air can be considered as ideal gas.

$$\frac{p}{\rho} = R_{air} T \tag{2.21}$$

## 2.2 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, which is based on the solution of the equations in integral form [1] [2]. Referring to a governing equation for generic property $\varphi$:

$$\frac{\partial}{\partial t}\int_\Omega \rho\varphi d\Omega + \int_S \rho\varphi \mathbf{u} \cdot \mathbf{n} dS = \int_S \Gamma\nabla\varphi \cdot \mathbf{n} dS \int_\Omega Q_\varphi \, d\Omega \tag{2.22}$$

where $\Gamma$ represents $\varphi$ diffusivity and $Q_\varphi$ 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

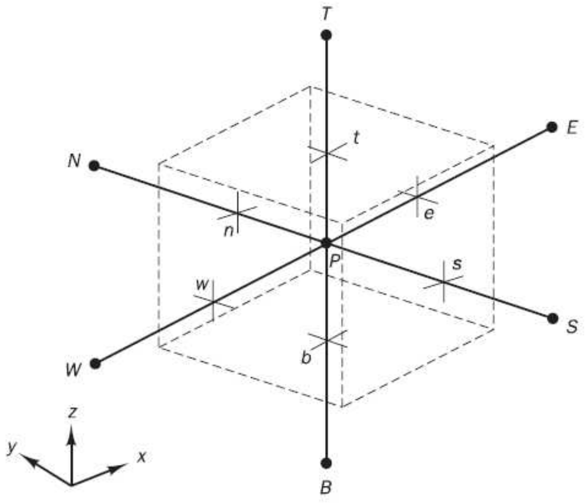


*Figure 2.1: A cell in three dimension and neighboring nodes*

### 2.2.1 Time discretization

This is a brief introduction of simple methods for temporal discretization. The time derivative term is usually discretized by means of a finite-difference ratio. The simplest scheme is the Euler one:

$$\frac{\partial \varphi}{\partial t} \approx \frac{\varphi^n - \varphi^{n-1}}{\Delta t} \tag{2.23}$$

where the superscript n denotes the time-step number and Δ t is the duration of the time step itself. Euler is a first order accuracy scheme. More sophisticated methods can be used, one of them is the backward differencing scheme (second order):

$$\frac{\partial \varphi}{\partial t} = \frac{3\varphi^n - 4\varphi^{n-1} + 4\varphi^{n-2}}{2\Delta t} \tag{2.24}$$

### 2.2.2 Space discretization

After defining the grid, the second step is to discretize the equations in order to solve the algebraic equation system. Flow through the volume face is defined as:

$$\int_S f dS = \sum_k \int_{S_k} f dS \tag{2.25}$$

Surface integral is equal to the sum of the integrals on each face of the control volume. $f$ is a generic function that refers to convective or diffusion term of equation (2.22). To approximate the integral for generic function f, a simple second order method called midpoint rule can be used, for which the integral of the k-th surface of f function is equal to the product of the integrand function at the center of surface k ($f_k$) by the area of the same surface:

$$\int_S f dS = f_k S_k \tag{2.26}$$

The value of f at the face center is not known, it must be interpolated from the value of the node, placed at the cell center. To ensure the same order of accuracy, interpolation must be at least second order. There are other interpolation methods with greater accuracy, that require knowledge of more than one point on the surface, but they are difficult to implement.

A similar procedure is carried out for volume integrals that appear in the equation (2.22). Also in this case, a second order approximation is sufficient and suitable. With midpoint rule the volume integral is equal to the product of the mean value of integrand function by the control volume.

$$\int_\Omega Q_\varphi d\Omega = Q_\varphi \Delta\Omega \tag{2.27}$$

In this case, since the mean value is placed on the center of control volume, then one does not need to interpolate. More accurate interpolation methods require knowledge of more values of Q in the volume and not only in the cell center and this is generally obtained with either interpolation or shape functions.

### 2.2.3 Equations discretization methods

Discretization of the governing equations [3] [4] allows to turn a system of partial differential equations into a system of algebraic equations that can be expressed as:

$$[A]\mathbf{x} = \mathbf{b} \tag{2.28}$$

Figure 2.2 shows the notation of geometric parameters; P and N are nodes in cell centers, $f$ is the common face to two cells and $\mathbf{S}_f$ is its normal. $\mathbf{d}$ indicates the distance between P and N and $\mathbf{d}_f$ is the distance between the common face f and the cell center N.

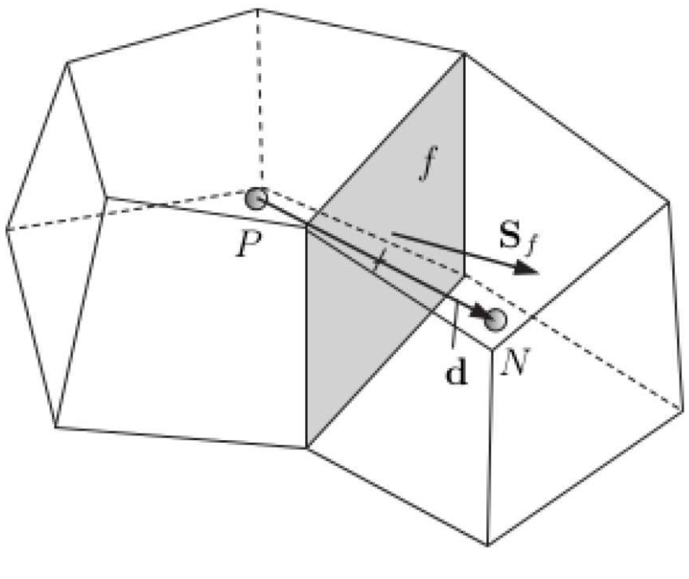


*Figure 2.2: Notation in the finite volume discretization*

Gradient can be discretized by applying the Gauss theorem:

$$\int_\Omega \nabla\varphi d\Omega = \int_S \varphi dS = \sum_f \mathbf{S}_f \varphi_f \tag{2.29}$$

If face-normal gradient is needed, it can be computed directly as a finite difference between cell-centered values:

$$(\nabla\varphi)_f = \frac{\varphi_N - \varphi_P}{|\mathbf{d}|} \tag{2.30}$$

In this way the gradient is measured along the line connecting the cells, it is linearly interpolated from the center to the cell face, but the accuracy is penalized if the mesh is non-orthogonal. In this case it is possible to introduce an additional explicit term that performs a correction, interpolating the gradients of the cell centers. Referring to Figure 2.3 it is possible to write:

$$(\mathbf{S} \cdot \nabla\varphi)_f = A(\varphi_N - \varphi_P + \mathbf{k}(\nabla\varphi)_f \tag{2.31}$$

with

$$A = \frac{|\mathbf{S}_f|^2}{\mathbf{S}_f \mathbf{d}} \tag{2.32}$$

and

$$\mathbf{k} = \mathbf{S}_f - A\mathbf{d} \tag{2.33}$$

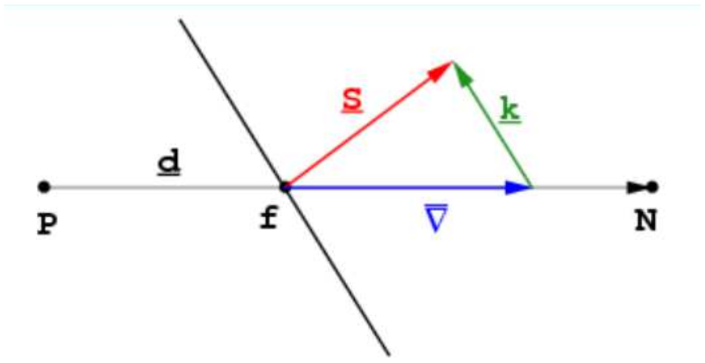


*Figure 2.3: Gradient correction for non-orthogonal mesh*

Divergence terms can be discretized by applying Gauss theorem:

$$\int_\Omega \nabla \cdot \varphi d\Omega = \int_s dS \cdot \varphi = \sum_f \mathbf{S}_f \cdot \varphi_f \tag{2.34}$$

Laplacian term is integrated on the volume and it can be discretized as:

$$\int_\Omega \nabla \cdot (\Gamma \nabla\varphi d\Omega) = \int_s dS \cdot (\Gamma \nabla\varphi d\Omega) = \sum_f \nabla_f \mathbf{S}_f \cdot (\nabla\varphi)_f \tag{2.35}$$

$\Gamma$ is the diffusive term. $\Gamma_f$ is linearly interpolated for orthogonal mesh, otherwise it is possible to apply a correction similar to the previous for the gradient.

### 2.2.4 Interpolation methods

Variables values are placed at the center of control volume and, in order to solve the governing equations, the values at face ceters are required in several places. *Upwind Difference Scheme* (UDS) is a very simple and first order interpolation method. The upwind differencing takes into account the flow direction when determining the value at a cell face: the convected value of $\Phi$ at a cell face is taken to be equal to the value at the upstream node. With reference to Figure 2.4, $\Phi_e$ can be assumed as :

$$\Phi_e = \begin{cases} \Phi_P & if(\mathbf{u} \cdot \mathbf{n})_e > 0 \\ \Phi_E & if(\mathbf{u} \cdot \mathbf{n})_e < 0 \end{cases}$$

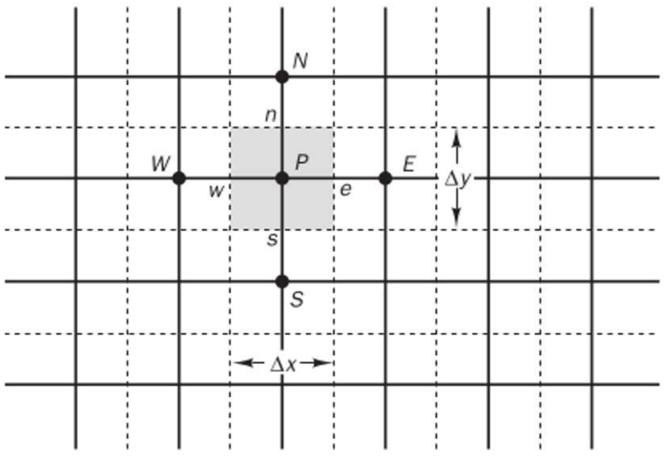


*Figure 2.4: Cartesian notation for a control volume in two dimensions*

The advantage of this scheme lies in the fact that it meets boundedness criterion and it does not generate oscillating solutions. This method introduces the numerical diffusion problem, however quite refined mesh can be used to obtain solution accuracy. This can be demonstrated with Taylor series expansions of $\Phi_e$ in the neighborhood of the point P:

$$\Phi_e = \Phi_P + (x_e - x_p)\left(\frac{\partial \Phi}{\partial x}\right)_P + (x_e - x_p)^2 \left(\frac{\partial^2 \Phi}{\partial x^2}\right)_P + O(\nabla x^3) \qquad (2.36)$$

Where $O$ $(\nabla x^3)$ are higher order terms. Upwind scheme is a first order approximation, so its truncation error depends on the second derivative, that is similar to diffusive flux of governing equation. The numerical diffusion coefficient $\Gamma_e^{num}$ can be defined as:

$$\Gamma_e^{num} = (\rho u)_e \frac{\Delta x}{2} \tag{2.37}$$

This factor has significant influence on the numerical solution, particularly in three-dimensional problems with oblique flow to the mesh, where, in addition to the diffusion along the flow direction, it generates a diffusive contribution also in the normal direction to it. This effect decreases with increasing grid refinement.

*Central Difference Scheme* (CDS) is a second order method. It consists in a linear interpolation between two cell centers in order to obtain face value request.

$$\Phi_e = \Phi_E \lambda_e + \Phi_P (1 - \lambda_e) \tag{2.38}$$

With $\lambda_e$ called interpolation factor and defined as:

$$\lambda_e = \frac{x_e - x_P}{x_E - x_P} \tag{2.39}$$

It is precise method even if less stable than upwind method. *Linear Upwind Difference Scheme* (LUDS) is a second order scheme and, unlike Upwind first order scheme, it evaluates a variable with linear extrapolation on two upstream nodes according to the direction of the fluid. With reference to Figure 2.4:

$$\Phi_e = \begin{cases} \frac{4\Phi_P - \Phi_W}{3} & if(\mathbf{u} \cdot \mathbf{n})_e > 0 \\ \frac{4\Phi_E - \Phi_{EE}}{3} & if(\mathbf{u} \cdot \mathbf{n})_e < 0 \end{cases}$$

This scheme is more accurate than a first order method; however it is possible that oscillatory solutions are generated.

### 2.2.5 Solution of linear equations systems

In this section the algorithms for solving linear systems are described in a concise way. For a more complete discussion, refer to [1, 3].

The discretized equations can be rewritten in a more compact matrix form as:

$$[A]\boldsymbol{\psi} = \mathbf{b} \tag{2.40}$$

where [A] is a relatively sparse matrix of known coefficient, $\boldsymbol{\psi}$ is a column vector of unknowns, and $\mathbf{b}$ is a column vector of known quantities.

For the solution of the system of linear equations such as equation (2.40), two main types of methods are used. The former are *direct methods.* They give the exact solution of the system after a number of operations. The latter are *iterative methods*, for which the solution is approximated by successive corrections. These two main types of methods are described below.

Direct methods allow to get the solution of the system without the explicit calculation of the inverse matrix of [A]. One of the most widespread methods is *Gauss Elimination.* Its basis is the systematic reduction of large systems of equations to smaller ones. In this procedure, the elements of the matrix are modified but the dependent variable names do not change. In the first phase, rows are properly combined in order to obtain an upper triangular matrix. This is the most computationally heavy step, proportional to $n^3/3$ where n x n is the matrix dimension. The phase that has just been described is called forward elimination. In the second phase the term contained in the last row is replaced in the rows above, in order to solve the equation in one unknown for all the rows and reach the solution.

Another direct method is *LU Decomposition.* Considering algebraic system (2.40) the matrix A can be factored into the product of lower [L] and upper [U] triangular matrices:

$$[A] = [L][U] \tag{2.41}$$

To make the factorization unique, it is required that the diagonal elements of L, lii, all be unity; alternatively, the diagonal elements of U could be required to be unity. The existence of this factorization allows the solution of the system of equation (2.40) in two stages:

$$[U]\boldsymbol{\psi} = \boldsymbol{y} \tag{2.42}$$

$$[L]\mathbf{y} = \mathbf{b} \tag{2.43}$$

The vector **y** is first calculated in equation (2.42) and then it is substituted in equation (2.43), in order to determinate the unknown vector $\boldsymbol{\psi}$. The advantage of LU factorization over Gauss elimination is that the factorization can be performed without knowing the vector **b**. As a result, if many systems involving the same matrix are to be solved, considerable savings can be obtained by performing the factorization first; the system can then be solved as requires.

Any system of equations can be solved by Gauss elimination or LU decomposition. Unfortunately, the triangular factors of sparse matrices are not sparse, so the cost of these methods is quite high. Furthermore, the discretization errors are usually much larger then the accuracy of the computer arithmetic so there is no reason to solve the system that accurately. Solution to somewhat more accuracy than that of the discretization scheme suffices.This leaves an opening for iterative methods. They are necessarily used for non-linear problems, but they are just as valuable for sparse linear system. In an iterative method, after a solution is guessed, the equation is used to systematically improve it. If each iteration is cheap and the number of iteration is small, an iterative solver may cost less than a direct method. In CFD problems this is usually the case. Iterative methods consist in the generation of a sequence of vectors $\boldsymbol{\psi}^m$ (m iteration index) that converges at the exact solution:

$$\lim_{m\to\infty} \boldsymbol{\psi}^m = \boldsymbol{\psi} \tag{2.44}$$

the recursive resolution scheme is:

$$\boldsymbol{\psi}^{m+1} = [B]\boldsymbol{\psi}^m + \mathbf{g} \tag{2.45}$$

where [B] and g are dependent on the [A] and q, they must satisfy the consistency condition:

$$\boldsymbol{\psi} = [B]\boldsymbol{\psi} + \mathbf{g} \tag{2.46}$$

Referring to equation (**??**), $\mathbf{g}$ is defined as:

$$\mathbf{g} = [I - B][A]^{-1}\mathbf{b} \tag{2.47}$$

the error after $m$ time step is:

$$e^m = \boldsymbol{\psi} - \boldsymbol{\psi}^m \tag{2.48}$$

and the residual:

$$\mathbf{r}^m = \mathbf{b} - [A]\boldsymbol{\psi}^m \tag{2.49}$$

Iterative methods generate a sequence of approximate solutions to the system that (hopefully) converge to the exact solution. After $m$ iterations, an approximation to the exact solution is obtained as:

$$[A]\boldsymbol{\psi}^m = \mathbf{b} - \mathbf{r}^m \tag{2.50}$$

The purpose of iteration procedure is to drive the residual to zero; in the process, also the error tends to zero. To see how this can be done, consider an iterative scheme for a linear system; such a scheme can be written:

$$[M]\boldsymbol{\psi}^{m+1} = [N]\boldsymbol{\psi}^{m} + \mathbf{B} \qquad (2.51)$$

An obvious property that must be demanded to an iterative method is that the converged result satisfies equation (2.40). Since, by definition, at convergence, $\boldsymbol{\psi}^{m+1} = \boldsymbol{\psi}^{m} = \boldsymbol{\psi}$ we must have:

$$[A] = [M] - [N] \quad \mathit{and} \quad \mathbf{B} = \mathbf{b}, \qquad (2.52)$$

or more generally,

$$[P][A] = [M] - [N] \quad \mathit{and} \quad \mathbf{B} = [P]\mathbf{b}, \qquad (2.53)$$

where [P] is a non-singular *pre-conditioning matrix.*

There are many iterative methods, but only those that have been actually used in this study will be described. The resolution with *Gauss-Seidel Method* requires as pre-conditioning matrix [P] the lower triangular part of [A]. This method converges if [A] is strictly row diagonally dominant or if it is symmetric positive definite matrix; it converges very quickly, twice as fast as the Jacobi method but this is not a sufficient improvement to be useful.

In this section, a class of methods based on techniques for solving nonlinear equations is presented. These methods begin by converting the original system of equations into a minimization problem. Suppose that the original system of equation to be solved is given by equation (2.40) and that the matrix [A] is symmetric and its eigenvalue is positive; such a matrix is called *positive definite.* For positive definite matrices, solving the system of equations (2.40) is equivalent to the problem of finding the minimum of:

$$F = \frac{1}{2}\boldsymbol{\psi}^{T} A \boldsymbol{\psi} - \boldsymbol{\psi}^{T}\mathbf{b}. \qquad (2.54)$$

The oldest and best method for seeking the minimum of a function consists in *steepest descents.* The function F may be thought as a surface in (hyper)-space. At that point, we find the steepest downward path on the surface; it lies in the opposite direction to the gradient of the function. We then search for the lowest point on that line. By construction, it has a lower value of F than the starting point; in this sense, the new estimate is closer to the solution. The new value is then used as the starting point for next iteration and the process continues until it converges. Unfortunately, although it is guaranteed to converge, steepest descents often converges very slowly because the method tends to use the same search directions over and over again. Many improvements have been suggested. The easiest ones require the new search directions to be as different as possible from the old ones.

*Conjugate Gradient Method* is based on a remarkable discovery: it is possible to minimize a function with respect to several directions simultaneously while searching in one direction at a time. This is possible thanks to a clever choice of directions. While the conjugate gradient method guarantees that the error is reduced on each iteration, the size of the reduction depends on the search direction. It is not unusual for this method to reduce the error only slightly for a number of iterations and then find a direction that reduces the error by an order of magnitude or more in one iteration. It can be shown that the rate of convergence of this method depends on the *condition number k* of the matrix where:

$$k = \frac{\lambda_{max}}{\lambda_{min}} \tag{2.55}$$

and $\lambda_{max}$ and $\lambda_{min}$ are the largest and the smallest eigenvalues of the matrix. Although the conjugate gradient method is significantly faster than steepest descents for a given conditioned number, this basic method is not very useful. This method can be improved by replacing the problem whose solution is sought by another one with the same solution but a smaller condition number. This is called preconditioning. One way to precondition the problem is to pre-multiply the equation by another matrix. Since this would destroy the symmetry of matrix, the preconditioning must take the following form:

$$[P]^{-1}[A][P]^{-1} = [P]^{-1}\mathbf{b} \tag{2.56}$$

In this description, $\mathbf{r}^k$ is the residual at the $k$th iteration, $\mathbf{p}^k$is the $k$th search direction, $\mathbf{z}^k$ is an auxiliary vector and $\alpha_k$ and $\beta_k$ are parameters used in constructing the new solution, the residual, and the search direction. The algorithm can be summarized as follows:

- Initialize by setting: k=0, $\boldsymbol{\psi}^0 = \boldsymbol{\psi}_{in}, \mathbf{r}^0 = \mathbf{b} - [A]\boldsymbol{\psi}_{in}, \mathbf{p}^0 = 0$:
- Advance the counter: k = k+1
- Solve the system: $[M]\mathbf{z}^k = \mathbf{r}^{k-1}$
- Calculate: $s^k = \mathbf{r}^{k-1}\mathbf{z}^k$

  $\beta^k = s^k/s^{k-1}$
  $\mathbf{p}^k = \mathbf{z}^k + \beta^k\mathbf{p}^{k-1}$
  $\alpha^k = s^k/(\mathbf{p}^k \cdot [A]\mathbf{p}^k)$
  $\boldsymbol{\psi}^k = \boldsymbol{\psi}^{k-1} + \alpha^k\mathbf{p}^k$
  $\mathbf{r}^k = \mathbf{r}^{k-1} - \alpha^k[A]\mathbf{p}^k$

- Repeat until convergence

This algorithm involves solving a system of linear equations at the first step. The matrix involved is $[M] = [P]^{-1}$, where [C] is the pre-conditioning matrix, which is in fact never actually constructed. For the method to be efficient, [M] must be easy to invert.

The conjugate gradient method previously described is applicable only to symmetric systems. In order to apply this method to systems of equations that are not necessarily symmetric, it is necessary to convert an asymmetric problem to a symmetric one. The following method, called *bi-conjugate gradient*, results:

- Initialize by setting: k=0, $\boldsymbol{\psi}^0 = \boldsymbol{\psi}_{in}, \mathbf{r}^0 = \mathbf{b} - [A]\boldsymbol{\psi}_{in}, \overline{\mathbf{b}} - [A]\boldsymbol{\psi}_{in}$, $\mathbf{p}^0 = \overline{\mathbf{p}^0} = 0$:
- Advance the counter: k = k+1
- Solve the system: $[M]\mathbf{z}^k = \mathbf{r}^{k-1}, [M]\overline{\mathbf{z}}^k = \overline{\mathbf{r}}^{k-1}$
- Calculate: $s^k = \mathbf{r}^{k-1}\mathbf{z}^k$

  $\beta^k = s^k/s^{k-1}$
  $\mathbf{p}^k = \mathbf{z}^k + \beta^k\mathbf{p}^{k-1}$
  $\overline{\mathbf{p}}^k = \overline{\mathbf{z}}^k + \beta^k\overline{\mathbf{p}}^{k-1}$
  $\alpha^k = s^k/(\overline{\mathbf{p}}^k[A]\mathbf{p}^k)$
  $\boldsymbol{\psi}^k = \boldsymbol{\psi}^{k-1} + \alpha^k\mathbf{p}^k$
  $\mathbf{r}^k = \mathbf{r}^{k-1} - \alpha^k[A]\mathbf{p}^k$
  $\overline{\mathbf{r}}^k = \overline{\mathbf{r}}^{k-1} - \alpha^k[A]^T\overline{\mathbf{p}}^k$

- Repeat until convergence

The second algorithm requires almost exactly twice as much effort per iteration as the standard conjugate gradient method but it converges in about the same number of iteration.

The final method for solving linear system to be discussed in this study is the *multi-grid method.* A multi-grid method, employing grids of different mesh size, allows to solve all wave-length components and provides rapid convergence rates. The multigrid strategy combines two complementary schemes. The high-frequency error components are reduced applying iterative methods like Jacobi or Gauss-Seidel schemes. For this reason, these methods are called smoothers. On the other hand, low frequency error components are effectively reduced by a coarse-grid correction procedure. Since the action of a smoothing iteration leaves only smooth error components, it is possible to represent them as the solution of an appropriate coarser system. Once this coarser problem is solved, its solution is interpolated back to the fine grid to correct the fine grid approximation for low-frequency errors.

## 2.3 RhoSimple Scheme

The computational code used is OpenFOAM. It provides different types of solvers as well as offering the possibility to modify or create them according to the user's needs. For this work *rhoSimpleFoam* solver has been used; it is a steady state solver for laminar or turbulent compressible flow.

The algorithm that has been employed differs from SIMPLE (*Semi Implicit Method for Pressure-Linked Equations*) used for incompressible flow, since it is necessary to solve the energy equation in addition to the continuity and momentum equations. The simulations are characterized by a high compression ratio and the geometries analyzed have narrow passages, such as the throat section of the valve seat. This aspect creates problems of stability of the SIMPLE algorithm which has to be modified to achieve the convergence. This is a brief introduction of SIMPLE algorithm, which, shares some steps with the rhoSimpleFoam algorithm, that will be shown later. In the Navier-Stokes equations it is possible to see that there is not an independent pressure equation, however it appears in the gradient form in the momentum equation. Therefore a relation defining a field pressure that satisfies the continuity equation is needed; it is called *Poisson equation.* The divergence is estimated from the momentum equation (2.9) and, taking into account that the density is constant through continuity equation, Poisson equations can be obtained as:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right] \tag{2.57}$$

where the outer derivative of the pressure is a divergence originated from continuity equation and the inner derivative is a gradient originated from momentum equation. Due to the presence of non-linear terms that depend on each other, the resolution of the pressure field cannot be made directly, but the *predictor-corrector* iterative method must be used. For steady state problems, it is preferred to use implicit schemes, because they have less stringent limits on the time step used and allow to achieve convergence faster than explicit schemes. For each generic m-iteration the discretized equation to solve is:

$$A_P^{u_i} u_{i,P}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = B_{u_i}^{m-1} - \left( \frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{2.58}$$

where P is the index of the compute node, l is the index of the center of the cell adjacent to that of the center P, B integrates source term and external forces but not the pressure, A is the matrix of coefficients of the algebraic system like $[A]\mathbf{u} = \mathbf{b} - \nabla \boldsymbol{p}$.

Velocity can be calculated as:

$$u_{i,P}^{m*} = \frac{B_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left( \frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{2.59}$$

or more compactly:

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left( \frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{2.60}$$

The velocity calculated on the derived pressure from the previous iteration is a predicted value and it must be corrected to satisfy the continuity equations. Correction is introduced as:

$$u_{i,P}^{m} = u_{i,P}^{m*} + u'_{i,P} \tag{2.61}$$

velocity corrected equation as the following is obtained:

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left( \frac{\delta p^{m}}{\delta x_i} \right)_P \tag{2.62}$$

where $p^m$ is the corrected pressure at the current iteration m and is corrected as:

$$p^m = p^{m-1} + p' \tag{2.63}$$

the corrected velocity is introduced in discretized continuity equation:

$$\left( \frac{\delta \rho u_i^m}{\delta x_i} \right)_P \tag{2.64}$$

the pressure correction $p'$ is obtained as follows:

$$\frac{\delta}{\delta x_i} \left[ \frac{\rho}{A_P^{u_i}} \left( \frac{\delta p'}{\delta x_i} \right) \right]_P = \left[ \frac{\delta (\rho u_i^{m*})}{\delta x_i} \right] + \left[ \frac{\delta (\rho \tilde{u}'_i)}{\delta x_i} \right] \tag{2.65}$$

The velocity correction term $\tilde{u}'_i$ is unknown, and in SIMPLE algorithm is neglected, whereas in other methods it is estimated. When the pressure correction term $p'$ is obtained, it can be substituted in (2.63) in order to calculate the pressure field at the current iteration. SIMPLE algorithm does not converge rapidly, because term $\tilde{u}_i'$ is neglected; in this situation instability is likely to be generated. It is useful to introduce under relaxation factor $\alpha_p$ after correction pressure value $p'$ is calculated. $p^m$ is rewritten as:

$$p^m = p^{m-1} + \alpha_p p' \tag{2.66}$$

In the same way an under relaxation factor for $\alpha_u$ is introduced in order to guarantee the stability of the solution. To solve a compressible flow problem it is necessary to solve not only continuity and momentum equations but also energy and state equations. This operation is required because in compressible flow the heat production by the viscous term can be considerable, as well as the importance of the conversion of kinetic energy into internal energy during the motion and vice versa. In this algorithm, density is obtained from continuity equation, temperature is calculated from energy equation and discretized momentum equation for $m$ th-iteration is modified compared to the previously presented one:

$$u_{i,P}^{m*} = \frac{B_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{\Delta\Omega}{A_P^{u_i}} \left( \frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{2.67}$$

As already mentioned in the SIMPLE for incompressible flow, $u_{i,P}^{m*}$ is a predicted velocity and it does not satisfy the continuity equation. Referring to control volume shown in Figure 2.1, the mass balanced through the face is calculated as:

$$\frac{(\rho^{m-1} - \rho^m \Delta\Omega)}{\Delta t} + \dot{m}_e^* + \dot{m}_w^* + \dot{m}_n^* + \dot{m}_s^* = B_m^* \tag{2.68}$$

density and velocity are based on the previous iteration values. The mass flow rate for $e$-face is estimated as:

$$\dot{m}_e^* = \int_{S_e} \rho u^* \cdot n \approx (\rho u^* \cdot n)_e S_e \tag{2.69}$$

In equation (2.68) $B_m^*$ does not allow to fulfill the continuity equation. In compressible problem, the mass flow is dependent on variable density and face normal component of the velocity $u_n$. In this case both variables must be corrected. The flow after correction at *m-th* iteration can be formulated as:

$$\dot{m}_e^m = (\rho^{m-1} + \rho')_e (u_n^{m*} + u_n')_e S_e \tag{2.70}$$

where $\rho'$ and $u_n'$ represent respectively the density and velocity correction. The mass flow correction is defined as:

$$\dot{m}'_e = (\rho^{m-1} S u'_n)_e + (u_n^{m*} S \rho')_e + (\widetilde{\rho' u'_n S})_e \tag{2.71}$$

The term marked with a tilde is higher order and tends to zero faster than the others; for this reason it can be neglected. In the first term on right hand side there is only velocity correction, while density is the calculated value at the previous iteration. The flow can be corrected defining the velocity correction through pressure gradient correction as already done for incompressible flow:

$$(\rho^{m-1} S u'_n)_e + (u_n^{m*} S \rho')_e = (\rho^{m-1} S \Delta\Omega)_e + \overline{\left(\frac{1}{A_P^{u_n}}\right)_e} \left(\frac{\delta p'}{\delta n}\right)_e \tag{2.72}$$

The coefficient $A_P^{u_n}$ is the same for each Cartesian component of the velocity, so it is possible to substitute $A_P^{u_n}$=$A_P^u$

The second term on right hand side is the result of compressibility, then it is necessary to express density correction as function of pressure correction term as for velocity. Solving energy equations, the temperature is known and considered fixed at each iteration; therefore, it is possible to calculate:

$$\rho' \approx \left(\frac{\partial \rho}{\partial p}\right)_T p' = C_\rho p' \tag{2.73}$$

where for ideal gas:

$$C_\rho = \left(\frac{\partial \rho}{\partial p}\right)_T = \frac{1}{RT} \tag{2.74}$$

It is possible to write the second term correction as:

$$(u_n^{m*} S \rho')_e = \left(\frac{C_\rho \dot{m}^*}{\rho^{m-1}}\right)_e p'_e \tag{2.75}$$

The corrected mass flow rate on the $e$ face can be rewritten as:

$$\dot{m}'_e = (\rho^{m-1} S \Delta\Omega)_e \overline{\left(\frac{1}{A_P^{u_n}}\right)_e} \left(\frac{\delta p'}{\delta n}\right)_e + \left(\frac{C_\rho \dot{m}^*}{\rho^{m-1}}\right)_e p'_e \tag{2.76}$$

The values of $p'$ at faces center and the gradient normal component at the faces are unknown, then they are interpolated as seen before in finite volume method.

The continuity equation with corrected terms can be rewritten as:

$$\frac{\rho'_P \Delta\Omega}{\Delta t} + \dot{m}_e + \dot{m}_w + \dot{m}_n + \dot{m}_s + B_m^* = 0 \tag{2.77}$$

Replacing density and flow correction term with the relation previously described, in function of pressure correction, the algebraic system of equations can be written as:

$$A_P p'_P + \sum_l A_l p'_l = -B^*_m \tag{2.78}$$

From this relation it is possible to obtain pressure correction value $p'_p$ for $m$th-iteration. This algorithm is repeated until the equations comply with a certain tolerance set. For under relaxation factor the same relation for incompressible flow is applied. In addition, even variables $\rho$ and $e$ or $h$ are under-relaxed.

## 2.4 Turbulence modeling

Most flows encountered in engineering practice are turbulent and therefore require different treatment. Turbulent flows are characterized by the following properties [5]:

- 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.

Before proceeding to the discussion of numerical methods for these flows, it is useful to introduce a classification scheme for the CFD approaches to predict turbulence flows.

- The method used for this work is based on equations obtained by averaging the equations of motion over time, over a coordinate in which the mean flow does not vary, or over an ensemble of realizations. This approach is called *one-point closure* and leads to a set of partial differential equations called the *Reynolds-averaged Navier-Stokes* (or RANS) equations. These equations do not form a closed set so this method requires the introduction of approximations (*turbulence model*).

- *Large eddy simulation (LES)* solves largest scale motions of the flow while approximating or modeling only the small scale motion.

- *Direct numerical simulation (DNS)* in which the Navier-Stokes equations are solved for all of the motions in a turbulent flow.

### 2.4.1 Reynolds Average Navier Stokes

In a statically steady flow, every variable can be written as the sum of a time averaged value and a fluctuation about that value:

$$\Phi(x_i, t) = \bar{\Phi}(x_i) + \Phi'(x_i, t) \tag{2.79}$$

where

$$\bar{\Phi}(x_i) = \lim_{T \to +\infty} \frac{1}{T} \int_0^T \Phi(x_i, t) dt \tag{2.80}$$

Here t is the time and T is the averaging interval. This interval must be large compared to the typical time scale of the fluctuations; thus, we are interested in the limit of T $\to +\infty$, Figure 2.5. If T is large enough, $\bar{\Phi}$ does not depend on the time at which the averaging is started.

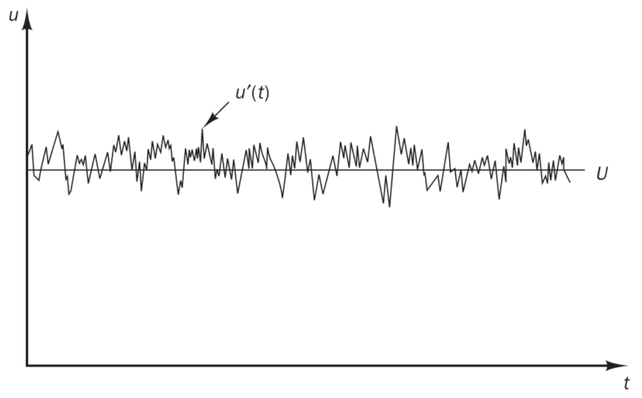


*Figure 2.5: Time averaging for a statically steady flow*

If the flow is unsteady, time averaging cannot be used and it must be replaced by ensemble averaging:

$$\bar{\Phi}(x_i, t) = \lim_{T\to+\infty} \frac{1}{N} \sum_{n=1}^{N} \Phi(x_i, t) \tag{2.81}$$

where N is the number of members of the ensemble and must be large enough to eliminate the fluctuations' effects. This type of averaging can be applied to any flow. We use the term *Reynolds averaging* to refer to any of these averaging processes; applying it to the Navier-Stokes equations, the Reynolds-averaged Navier-Stokes (RANS) equations are obtained. From equation (2.80), it follows that $\bar{\Phi'} = 0$. Thus, averaging any linear term in the conservation equations simply gives the identical term for the averaged quantity. From a quadratic nonlinear term we get two terms, the product of the average and a covariance:

$$\overline{u_i \Phi} = \overline{(\bar{u_i} + u_i')(\bar{\Phi} + \Phi')} = \bar{u_i}\bar{\Phi} + \overline{u_i'\Phi'} \tag{2.82}$$

The last term is zero only if the two quantities are uncorrelated; this is rarely the case in turbulent flow and, as a result, the conservation equations contain terms such as $\overline{u_i'\Phi'}$, known as the *turbulent scalar flux,*. These cannot be represented uniquely in terms of the mean quantities.

For incompressible flows without body forces, the averaged continuity and momentum equations can be written in tensor notation and Cartesian coordinates as:

$$\frac{\partial(\rho\bar{u}_i)}{\partial x_i} = 0 \tag{2.83}$$

$$\frac{\partial(\rho\bar{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\bar{u}_i\bar{u}_j + \rho\overline{u_i u_j}\right) = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial\bar{\tau_{ij}}}{\partial x_j} \tag{2.84}$$

where $\bar{\tau}_{ij}$ are the mean viscous stress tensor components:

$$\bar{\tau}_{ij} = \mu\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right) \tag{2.85}$$

The presence of the Reynolds stress and turbulent scalar flux in the conservation equations means that the latter are not closed, i.e. they contain more variables than equations. Closure requires the use of some approximations, that follow Reynolds stress approximations, which, in turn usually take the form of the mean quantities.

It is possible to derive equations for the higher order correlations, e.g. for the Reynolds stress tensor, but they still contain more (and higher-order) unknown correlations that require modeling approximations. These equations will be introduced later but the important point is that it is impossible to derive a closed set of exact equations. The approximations introduced are called *turbulence models* in engineering. To close the equations it is necessary to introduce a turbulence model. To see what a reasonable model might be, it can be observed that in laminar flows, energy dissipation and transport of mass, momentum, and energy normal to the streamlines are mediated by the viscosity, so it is natural to assume that the effect of turbulence can be represented as an increased viscosity. This leads to the eddy-viscosity model for the Reynolds stress:

$$\rho\overline{u_i u_j} = \mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho\delta_{ij}k \tag{2.86}$$

and the eddy-diffusion model for a scalar:

$$\rho\overline{u_j\phi} = \Gamma\frac{\partial\overline{\phi}}{\partial x_j} \tag{2.87}$$

equation (2.86) $k$ is the turbulent kinetic energy:

$$k = \frac{1}{2}\overline{u_i' u_i'} \tag{2.88}$$

The last term equation (2.86) is required to guarantee that, when both sides of the equation are contracted, the equation remains correct. Although the eddy-viscosity hypothesis is not correct in detail it is easy to implement and, with careful application, can provide reasonably good results for many flows.

### 2.4.2 k-omega SST

The SST k-$\omega$ turbulence model [6] is a two-equation eddy-viscosity model. The first transported variable is turbulent kinetic energy $k$. The second transported variable in this case is the specific dissipation $\omega$. It is the variable that determines the scale of the turbulence, whereas the first variable, $k$, determines the energy in the turbulence. The shear stress transport (SST) formulation combines the best of two worlds. The use of a k-$\omega$ formulation in the inner parts of the boundary layer makes the model directly usable all the way down to the wall through the viscous sub-layer, hence the SST k-$\omega$ model can be used as a Low-Re turbulence model without any extra damping functions. The SST formulation also switches to a k-$\epsilon$ behavior in the free-stream and thereby avoids the common k-$\omega$ problem that the model is too sensitive to the inlet free-stream turbulence properties. Authors who use the SST k-$\omega$ model often merit it for its good behavior in adverse pressure gradients and separating flow. The SST k-$\omega$ model does produce a bit too large turbulence levels in regions with large normal strain, like stagnation regions and regions with strong acceleration. This tendency is much less pronounced than with a normal k-$\epsilon$ model. The kinematic eddy viscosity is defined as:

$$\nu_T = \frac{a_1 k}{max(a_1\omega, SF_2)} \tag{2.89}$$

The turbulent kinetic energy can be expressed as:

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \beta^* k\omega + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_k \nu_T)\frac{\partial k}{\partial x_i}\right] \tag{2.90}$$

The specific dissipation rate:

$$\begin{aligned} &\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega \nu_T)\frac{\partial \omega}{\partial x_i}\right] \\ &+ 2(1 - F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\partial\omega\partial x_i \end{aligned} \tag{2.91}$$

For the closure coefficients and auxiliary relations refer to [6]

# Chapter 3

# Experimental Tests

The laboratory of fluid dynamics at Centro Ricerche Fiat is able to perform several kinds of tests concerning the fluid dynamics characterization of different systems, measuring mass flow rate and pressure drop thus providing fundamental parameters for the design. The intake systems are the most tested devices. These experimental tests evaluate the permeability and the ability to produce organized motions within the cylinder. The purpose of these tests is to provide the curves of the discharge coefficient or tumble coefficient as a function of valve lift, and to determine whether the test configuration can lead to an improvement of the ability to produce organized motion without compromising the discharge coefficient and thus the permeability.

All tests are performed under steady state conditions and in the absence of combustion, i.e. in conditions significantly different from those occurring during engine operation. These results can be considered reliable only if they meet the following assumptions:

- The unsteady flow behavior can be approximated as a succession of steady states, one for each valve lift.
- Tests are carried out considering a pressure ratio between upstream and downstream of $\beta = 1.1$.
- The pressure drop is assumed as constant during the intake stroke.
- The intake flow occurs during the entire valve opening and it is determined only by the valve lift.

These assumptions may seem restrictive, however, they are the basis to ensure a robust test work. The second assumption imposes a strong constraint because in-cylinder pressure is variable over time, due to the piston motion, that does not appear in steady state flow tests.

To understand the measurements and the methodology used in these tests, first of all, it is necessary to describe the structure of the workbench. In these tests a blow bench has been used as it is shown in Figure 3.1. It works with an upstream pressure higher than the downstream pressure which is equal to the ambient pressure.

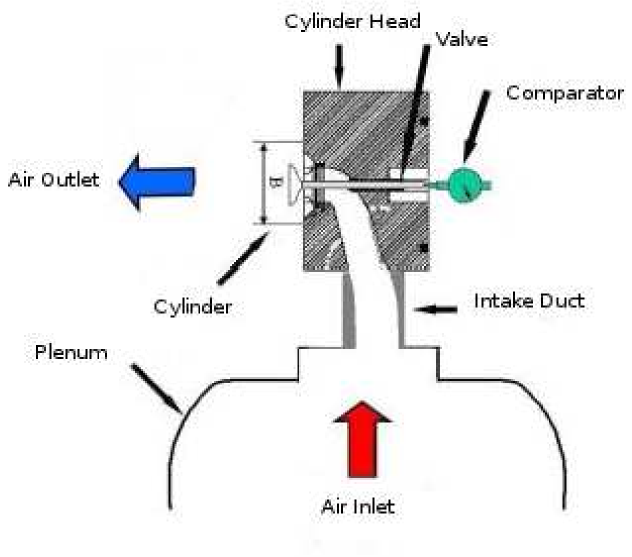


*Figure 3.1: Flow bench scheme*

There is a plenum chamber that dampens the fluctuations of the compressor, in order to ensure a steady state pressure level. The plenum must contain a volume of air much greater than other objects, so as to consider the average velocity of the flow negligible. The experimental results that are obtained with these tests are steady state type and then several tests with different valve lift are needed to characterize the fluid dynamics of the engine ducts. The valves are moved through valve-push, the valve lift is checked by using a dial gauge indicator placed on the top of stem of the valve-push.

The equipment used for the experimental tests consist in a flow box with symmetrical intake ducts. The tests are carried out on a blown bench with expansion ratio equal to $\beta = 1.1$. The maps of the velocity fields within the cylinder that are required to calculate the tumble number are obtained as a function of the valve lift. The flow straightener shown in Figure 3.2 has a thickness equal to 8 mm, and it is placed at the end of the cylinder. The cylinder has a length equal to half the bore.

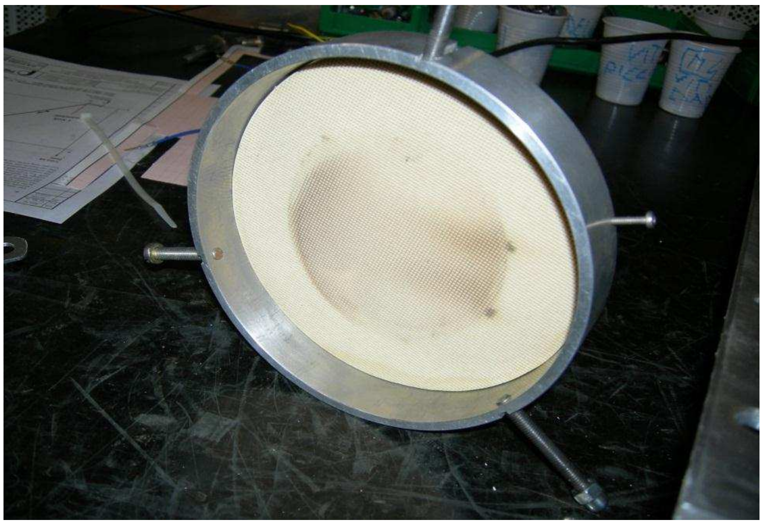


*Figure 3.2: Linearized*

The flow straightener is needed because the probe is able to measure only the modulus of the velocity and any information about direction is lost. The distance between the probe and the flow straightener is 5 mm, in order to avoid any collision between the two devices.

The probe consists of an Hot-Wire Anemometer. The core of the anemometer is an exposed hot wire either heated up by a constant current or maintained at a constant temperature. In either case, the heat lost due to fluid convection is a function of the fluid velocity. By measuring the change in wire temperature under constant current or the current required to maintain a constant wire temperature, the heat loss can be obtained. The lost heat can then be converted into a fluid velocity in according to heat transfer law.

a complete test with a matrix with length 100 mm and a pitch between the columns of 0.5 mm lasts for approximately 45 minutes. At the end of data acquisition the program allows to store the matrix of the voltage values in text file, in order to make them available for post processing analysis. The elaboration program is written in *Matlab* and requires the following data as input:

- Text file with the raw data matrix $[V]$.
- Text file containing the polynomial coefficient of probe calibration

curve [$m/sV$].

- Atmospheric pressure [$Pa$].
- Ambient temperature [$^oC$].
- Plenum temperature [$K$].
- Plenum pressure [$Pa$].
- Cylinder bore [$mm$].
- Length and height of the area investigated [$mm$].
- The sampling frequency [$Hz$].
- The speed in the downward movement of the probe during the acquisition.
- The pitch between the columns.

The program automatically calculates the number of rows, columns and the position of the center of the matrix according to the rows and columns coordinates, and it creates a velocity matrix from raw data and displaying it in the form of color map. The results of experimental tests will compared against simulation shown in Chapter 6.

# Chapter 4

# Simulation Setup

The *OpenFOAM* basic case directory structure, that contains the set of files required to run application, is shown in Figure 4.1.

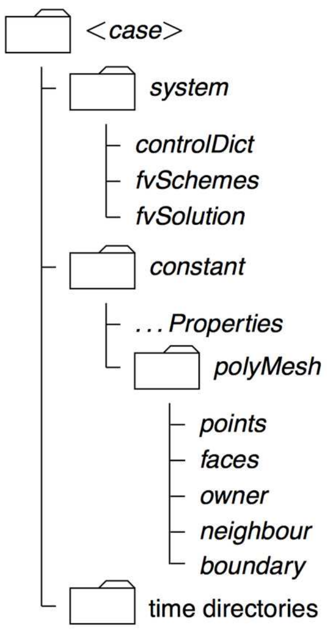


Figure 4.1: *OpenFOAM* directory structure

The *constant* directory contains a full description of the case mesh in a subdirectory polyMesh and dictionaries where setting thermophysical properties and turbulence models.

The *system* directory contains settings for the run. It must contain at least 3 files: *controlDict* where run control parameters are set including start/end time, time step and parameters for data output; *fvSchemes* where discretization schemes used in the solution may be selected at run-time; *fvSolution* which contains instructions to solve each discretized linear equation system, tolerances and other algorithm controls.

The *time* directory contains the dimension as well as the initial and the boundary condition for all variables. The initial conditions are usually stored in a directory 0.

## 4.1 Boundary and initial conditions

With the aim of solving a partial differential equations system, boundary and initial conditions must be properly assigned. All simulations were made in two steps. In the first step pressure drop coefficient = 1.1 is fixed as boundary condition in order to find the mass flow rate. In this case the total pressure for the inlet patch condition and static pressure for the outlet (piston) patch are imposed (BC1), as shown in Table 4.1. In the second step the mass flow rate, that has been found in previous simulation, is used as inlet boundary condition, keeping the same to the outlet boundary (BC2) Table 4.2. The second setting of boundary condition allows to obtain a lower time step continuity error, a better convergence on the residuals and generally a better stability for the first iterations.

| quantities | inlet | piston(outlet) | wall |
|---|---|---|---|
| $p$ | totalPressure | fixedValue | zeroGradient |
| $U$ | pressureInletVelocity | pressureInletOutletVelocity | noSlip |

Table 4.1: Boundary condition 1 (BC1)

| quantities | inlet | piston(outlet) | wall |
|---|---|---|---|
| $p$ | zeroGradient | fixedValue | zeroGradient |
| $U$ | flowRateInletVelocity | pressureInletOutletVelocity | noSlip |

Table 4.2: Boundary condition 2 (BC2)

For turbulent quantities, the k - $\omega$ model has been considered. Their boundary conditions have been suggested by reference to [9] [10].

$$k = \frac{3}{2}\overline{u'^2} \tag{4.1}$$

where $u' = 0.05U$

$$\omega = \frac{C_\mu^{3/4} k^{1/2}}{l} \tag{4.2}$$

where $l$ is the mixing length, that is defined as:

$$l = 0.079 \left( \frac{1}{5} D \right) \tag{4.3}$$

This setting procedure, especially at low valve lift (1 mm, 2 mm, 3 mm), led to a simulation stop in early time step. For medium and high valve lift (4 mm to 9 mm), the simulation has reached the solution without problems. After that, as previously described, the new boundary conditions have been updated with the mass flow rate and the new $k$, $\omega$ values. These new values have been calculated with an utility command: *patchAverage omega piston -time 6000* and *patchAverage k piston -time 6000* launched on previous simulation. This command allows quantities estimation on *piston* (outlet) patch at the end of the simulation (6000 is the maximum number of time step).

With the aim of setting $k$, $\omega$ boundary condition at lower lift, a linear extrapolation is performed with the linear interpolation mass flow rate value. Then knowing the exact $k$, $\omega$ and mass flow rate values at 4 mm and 5 mm valve lifts, and mass flow rate interpolated value at 3 mm valve lift, the values of $k$, $\omega$ at 3 mm valve lift can be calculated as:

$$\frac{k_4 - k_3}{\dot{m}_4 - \dot{m}_3} = \frac{k_5 - k_4}{\dot{m}_5 - \dot{m}_4} \tag{4.4}$$

$$\frac{\omega_4 - \omega_3}{\dot{m}_4 - \dot{m}_3} = \frac{\omega_5 - \omega_4}{\dot{m}_5 - \dot{m}_4} \tag{4.5}$$

This procedure is repeated also for 1 mm and 2 mm valve lift.

Near the wall $k$ - $\omega$ wall functions are used. Not being tied to a limited range of $y^+$ automatic wall treatments can be use. The boundary condition for turbulence quantities can be schematized in Table 4.3.

| quantities | inlet | piston(outlet) | wall |
|---|---|---|---|
| $k$ | inletOutlet | inletOutlet | compressible::kqRWallFunction |
| $\omega$ | inletOutlet | inletOutlet | compressible::omegaWallFunction |

*Table 4.3: Turbulence boundary condition*

In these cold simulations heat transfer is not so relevant. Temperature has been set equal to ambient temperature = 303 K for both boundary condition and internal field, moreover adiabatic cylinder head walls were considered.

| quantities | inlet | piston(outlet) | wall |
|---|---|---|---|
| $T$ | fixedValue | inletOutlet | zeroGradient |

*Table 4.4: Temperature boundary condition*

The whole process can be summarized in Figure 4.3

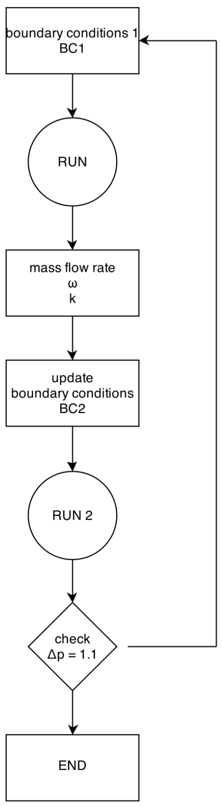


*Figure 4.2: Process scheme*

## 4.2 Numerical schemes

The $fvSchemes$ dictionary in the $system$ directory sets the numerical schemes for terms, such as derivatives in equations, that appear in applications being run. The first time derivative ($\frac{\partial}{\partial t}$) terms are specified in the $ddtSchemes$ sub-dictionary. In this work $staedyState$ scheme is used; it does not solve for time derivatives being a steady state flow simulation.

The *gradSchemes* sub-dictionary contains gradient terms. *cellLimited Gaus linear* and *cellMDLimited Gaus linear* discretizations schemes have been chosen. The former is used in first simulation with the BC1 and it guarantees more stability. The latter is used in second simulations with BC2 and it is more accurate. The *Gauss* keyword specifies the standard finite volume discretization of Gaussian integration which requires the interpolation of values from cell centers to face centers. Therefore, the Gauss entry must be followed by the choice of interpolation scheme; usually, as in this case, *linear* is employed. Gradient limiters have been used, in order to avoid oscillation on the gradient computations. The difference between gradient limiters is that *cellLimited* clips each component of the gradient equally, whereas *cellMDLimited* is a multi-dimensional limiter, whereby the gradient is clipped in the direction normal to the cell faces.

The *divSchemes* sub-dictionary contains divergence terms. The *Gauss* scheme is the only choice of discretization and requires a selection of the interpolation scheme for the dependent field. For divergence terms as *U, p, k,* $\omega$, *Upwind* first order bounded interpolation schemes can be chosen. For diffusive of the stress tensor *linear* is used.

The *laplacianSchemes* sub-dictionary contains Laplacian terms. Also in this case the *Gauss* scheme is the only choice of discretization and requires a selection of both an interpolation scheme for the diffusion coefficient and a surface normal gradient scheme. The unbounded, second order, conservative with non-orthogonal correction *Gauss linear corrected* scheme is used.

The *snGradSchemes* sub-dictionary contains surface normal gradient terms. A surface normal gradient is evaluated at a cell face; it is the component, normal to the face, of the gradient of values at the centers of the two cells that the face connects. The explicit non-orthogonal correction, *corrected*, is used.

## 4.3 Solution and algorithm control

The equation solvers, tolerances and algorithms are controlled by the $fvSolution$ dictionary in the *system* directory. The first sub-dictionary that appears in this dictionary is $solvers$. It specifies each linear solver that is used for each discretized equation. For pressure, $PCG$ (preconditioned conjugate gradient) linear solver is used. This solution method uses $diagonalincomplete - Cholesky$ as preconditioner option and $10^{-8}$ as tolerance. Velocity uses *smoothSolver* as linear solver and *GaussSeidel* as preconditioner. The number of sweeps must be specified, by the *nSweeps* keyword, before the residual is recalculated, following the tolerance parameters. *nSweeps* is set equal to 2 and tolerance is set equal to $10^{-7}$. For kinetic turbulent energy, turbulent dissipation rate, density and enthalpy, *PbiCG* (preconditioned bi-conjugate gradient) linear solver is used with *DILU* (Diagonal incomplete-LU) as preconditioner. Also in this case the tolerance is set equal to $10^{-7}$.

Since the mesh has non-orthogonality value less than 60, *nNonOrthogonalCorrectors* is set equal to 1. *residualControl* stops running if specified residual level is achieved on a given field. This setting is reported in Table 4.5.

| quantities | $residualControl$ |
|---|---|
| $p$ | $10^{-4}$ |
| $U$ | $10^{-4}$ |
| $h$ | $10^{-4}$ |
| $k$ | $10^{-5}$ |
| $\omega$ | $10^{-5}$ |

*Table 4.5: Residual control*

The $relaxationFactor$ in sub-dictionary controls under-relaxation, a technique used to improve stability of a computation, particularly in solving steady-state problems as in this case. Under-relaxation works by limiting the amount by which a variable changes from one iteration to the next, either by modifying the solution matrix and source prior to solving for a field or by modifying the field directly. The under-relaxation factors used in these simulations are defined in Table 4.6.

| quantities | $residualControl$ |
|---|---|
| $p$ | 0.3 |
| $rho$ | 0.03 |
| $U$ | 0.5 |
| $h$ | 0.3 |
| $k$ | 0.5 |
| $\omega$ | 0.5 |

*Table 4.6: Under-relaxation factors*

## 4.4 Flow straightener modeling

To model the flow straightener a porous medium can be used. This is a smarter choice than building a mesh, because the mesh would require a lot of cells and consequently high computational hardware resources. With porous media it is possible to replicate the straightener effect setting Darcy Forchheimer coefficients in $fvOption$ dictionary, contained in *system* directory. In order to block the flow movement in $x$ and $y$ direction, the Darcy Forchheimer coefficients, in the same directions, must be very high and equal to zero in $z$ direction. It is placed at half bore (40 mm) from the top of the cylinder head, in the same position of the flow straightener. Porous media is shown in Figure 4.3.

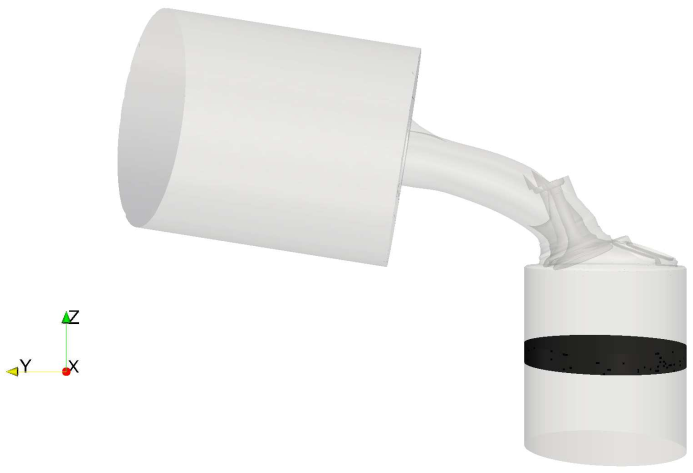


*Figure 4.3: Porous media within the cylinder*

# Chapter 5

# Mesh Generation

## 5.1 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.

The complex intake system shown in Figure 5.1, provided as CAD file, has required many tests in order to achieve a satisfactory simulation. In Figure 5.2 it is possible to see plenum and cylinder have been added. The former must contain a volume of air much greater than other objects, in order to ensure atmospheric condition and to consider the average velocity of the flow negligible. The latter shows a cylinder longer than experimental tests, which does not affect the results of simulation and stability is ensured.

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.

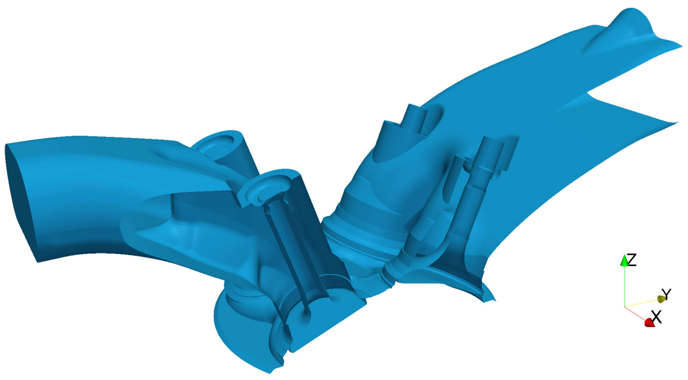


*Figure 5.1: Initial configuration of cylinder head*

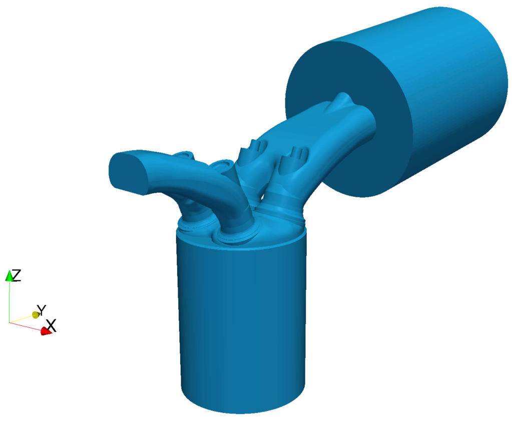


*Figure 5.2: Steady state flow configuration of cylinder head*

- Skewness.
- Aspect Ratio.
- Smoothness.

Referring to Figure 5.3 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.

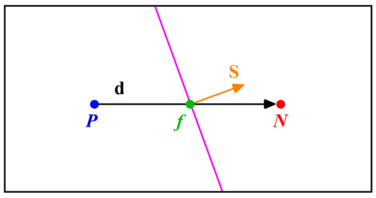


*Figure 5.3: Mesh orthogonality quality metrics*

$$i_{face,orth} = \frac{\boldsymbol{d} \cdot \boldsymbol{\Delta}}{|\boldsymbol{d}| \cdot |\boldsymbol{\Delta}|} \tag{5.1}$$

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

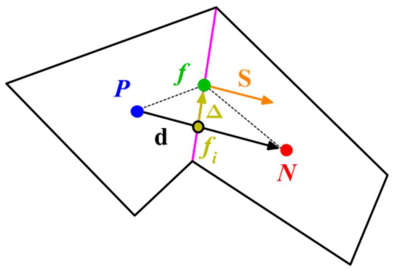


*Figure 5.4: Mesh skewness quality metrics*

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 $\boldsymbol{\Delta}$ and $f_i$ is the point where the vector **d** intersects the face $f$.

$$i_{face,skewness} = \frac{\boldsymbol{\Delta}}{\boldsymbol{d}} \tag{5.2}$$

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

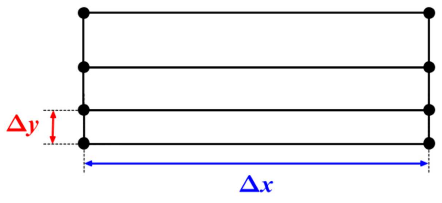


*Figure 5.5: Mesh aspect ratio quality metrics*

Mesh aspect ratio AR is the ratio between the longest side $\Delta x$ and the shortest side $\Delta y$

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.

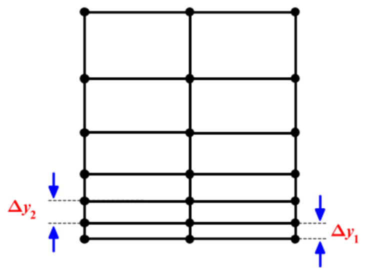


*Figure 5.6: Mesh smooth transition*

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

$$\frac{\Delta y_2}{\Delta y_1} = 1.2 \tag{5.3}$$

## 5.2 Mesh generation with the *snappyHexMesh*

### 5.2.1 BlockMesh

The first step is to create a background mesh of hexahedral cells that fills the entire region within the external boundary. This can be done using *blockMesh* utility supplied with OpenFOAM. It is not necessary to create a fine background mesh, since it will be refined by *snappyHexMesh*. The background mesh shown in Figure 5.7 is generated from a dictionary file named blockMeshDict, wherein vertices are defined:

```
convertToMeters 1;
vertices
(
    (-0.074 -0.105 -0.11)
    (0.038 -0.105 -0.11)
    (0.038 0.217 -0.11)
    (-0.074 0.217 -0.11)
    (-0.074 -0.105 0.12)
    (0.038 -0.105 0.12)
    (0.038 0.217 0.12)
    (-0.074 0.217 0.12)
);
blocks
(
hex(0 1 2 3 4 5 6 7)(56 161 115)simpleGrading(1 1 1)
);
```

The initial cell length is 2mm.

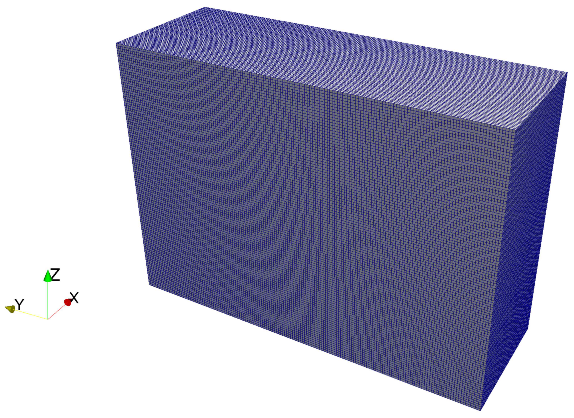


*Figure 5.7: Background mesh*

For optimum behavior cells should be close to unit aspect ratio.

For meshing the geometry, the mesh generation utility, *snappyHexMesh* supplied with OpenFOAM, can be used. *snappyHexMesh* utility generates 3D meshes containing hexahedral and split-hexaedral automatically from a triangulated surface geometry in Stereolithography (STL) format. The starting mesh is refined by iterative refinement up to a mesh approximately compliant to the surface, later the resulting split-hex mesh is morphed to the surface. The boundary cell layers are added in the final phase.

The dictionary file *snappyHexMeshDict* consists of six main sections, due to the large number of options which control the behavior of *snappyHexMesh*.

### 5.2.2 Basic controls and geometry

Basic controls section contains the file header and the keyword to switch on/off the different mesh step.

```
castellatedMesh true;
snap            true;
addLayers       true;
```

Geometry section contains the name of the input geometries, the definition of geometry *patch* previously divided into STL file and it also defines refinements zone and shape.

```
geometry
{
    lift_0.005.stl
    {
    type triSurfaceMesh;
     name headCRF;
         regions
         {
        cylinderHead
        { name cylinderHead; }
         coronaCircolare
        { name coronaCircolare;  }
        ...
        { name ...; }
        serbatoio
        { name serbatoio; }
        }
    }
        refinementBox
        {
        type searchableBox;
        min (-0.074 -0.063 -0.12);
        max (0.038 0.025 0.0113);
    }
};
```

### 5.2.3 CastellatedMesh

CastellatedMesh prescribes the first meshing stage, which is called refinement. In this step the background mesh is refined on the basis of surface and volume refinement settings.

```
castellatedMeshControls
{
    maxLocalCells 1000000;
    maxGlobalCells 4000000;
    minRefinementCells 1;
    nCellsBetweenLevels 3;
    features
        (
        );
// Surface based refinement
    refinementSurfaces
    {
        headCRF
        {
        level (2 3);
        regions
                {
                inlet
                {level (0 0);  }
                piston
                {level (1 1);  }
                liner
                {level (1 1);  }
                serbatoio
                 {level (0 0);  }
                }
        }
    }
    resolveFeatureAngle 30;
    refinementRegions
    {
        headCRF
        {
            mode distance;
            levels ((1.0 0));
        }
        refinementBox
        {
            mode inside;
            levels ((1E15 1));
        }
    }
    locationInMesh (-0.02 0.003 -0.025);
    allowFreeStandingZoneFaces true;
}
```

The main control parameters of castellated section are:

- *maxGlobalCells* defines a control over the global mesh size. The refinement will stop when this number of cell is reached. *maxlocalCells*, instead, defines a control over the maximum number of cells maintained per processor.

- *minRefinementCells* avoids to run too many refinement iterations on a limited number of cells. This setting will cause refinement to stop if number of cells to be refined $\leq$ minimumRefine. It was chosen equal to 1 because higher value led to an increase of computing time with an increase in cell quality.

- *nCellsBetweenLevels* command sets the number of buffer layers between different levels of refinement; even in this case a compromise must be found between the computing time and mesh refinement.

- *Surfacebasedrefinement* specifies two refinement levels for every surface Figure 5.8. The first is the minimum level: every cell intersecting a surface gets refined up to the minimum level. The second level is the maximum level of refinement. When the local curvature of the surface produces an intersection between the cell with an angle  resolveFeatureAngle, the maximum level of refinement is applied, as shown in Figure 5.9.

- *locationInMesh* command sets the Cartesian point to retain required volume mesh: set a point inside the stl geometry for an internal mesh and an outside point for external mesh. This point should never be on a face, always inside a cell. In this mesh an internal point has been chosen.

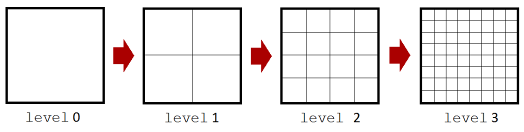


*Figure 5.8: succession mesh refinements*

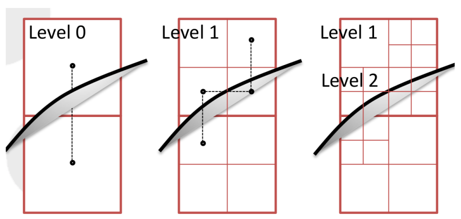


*Figure 5.9: Feature angle control*

### 5.2.4 Surface snapping

Surface snapping prescribes the second meshing step, which is called "snapping", where patch faces are projected down onto the surface geometry. The process performed can be summarized as:

- displace the vertices in the castellated boundary onto the STL surface;
- solve for relaxation of the internal mesh with the latest displaced boundary vertices;
- find the vertices that cause mesh quality parameters to be violated;
- reduce the displacement of those vertices from their initial value (at first point) and repeat from second point until mesh quality is satisfied.

```
// Settings for the snapping.
snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 100;
    nRelaxIter 5;
        nFeatureSnapIter 1;
        implicitFeatureSnap true;
        explicitFeatureSnap false;
        multiRegionFeatureSnap false;
}
```

The main control parameters of snap section are:

- *nSmoothPatch* command sets the number of pre-smoothing iterations of patch points before projection to the surface is performed.
- *tolerance* scaling the local maximum edge length to define the relative distance for points attraction to the surface. The greater the value the easier will be to find the correspondence. However the accuracy will be smaller.
- *nSolveIter* command sets the number of interior smoothing iterations applied to snapped displacement field. With a large number of cells is a sensitive parameter for time-consumption.
- *nRelaxIter*command intervenes after the snapping iteration, when a check of mesh is performed. If it fails, this parameter controls the number of scaling back iterations for error reduction stage. The amount of scale displacement is defined in *meshQualityControls* section and will be examined later.
- *nFeatureSnapIter* command checks the number of snapping iterations to perform over the feature edges and it leads to a better capture of the body edges, with a general improvement of the mesh validity.

### 5.2.5 Layer addition

In mesh layers section the final meshing stage is presented, which is called *layer addition*. In this final meshing stage, a layer of cells is added to a specified set of boundary patch. This is an optional stage of the meshing process, that introduces additional hexahedral cells aligned to the boundary surface. The methodology of mesh layer addition involves shrinking the existing mesh from the boundary and inserting layers of cells. The steps are the following:

- The mesh is pushed away from the surface by a specified thickness on the normal direction to the surface.
- Solve for relaxation of the internal mesh with the latest displaced boundary vertices.
- Check if validation criteria are satisfied, scale back the displacement if error occurs, reducing the projected thickness.

- If validation criteria can be satisfied insert mesh layers.
- Global mesh quality check: if check fails, layers are removed and the process restarts.

With this stage it has been possible to replace some irregular cells along boundary surface generated in snap stage, with hexahedral cells aligned to the boundary surface patch. This procedure is controlled by the settings in the *addLayersControls* sub-dictionary: it starts with the specification of the number of layers to be added on each patch.

```
{
// Settings for the layer addition.
addLayersControls
{
    relativeSizes true;
    layers
    {
        //cylinderHead
        //{ nSurfaceLayers 3; }
        coronaCircolare
        { nSurfaceLayers 3;  }
         ...
        { nSurfaceLayers 3;  }
        liner
        { nSurfaceLayers 3;  }
    }
    expansionRatio 1.15;
    finalLayerThickness 0.4;
    minThickness 0.1;
    nGrow 0;
    featureAngle 350;
    nRelaxIter 8;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 1;
    nSmoothThickness 1;
    maxFaceThicknessRatio 1.0;
    maxThicknessToMedialRatio 1.0;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 200;
    // nRelaxedIter 20;
}
```

The main control parameters of layer section are:

- *relativeSizes* command sets if the final layer thickness and the minimum thickness have to be defined as relative (*true*) to the background spacing, or defined as absolute (*false*) length.

- *expansionRatio* command controls the ratio of heights from one layer to the next consecutive layer in direction away from the surface.

- *finalLayerThickness* command sets the ratio of the final layer height relative to the adjacent surface mesh size.

- *minThickness* is the specification of a minimum layer thickness below which height layers will automatically be collapsed.

- *nRelaxIter* command controls the number of scaling back iterations during the error reduction stage, as seen in the "snapping" section.

- *featureAngle* command specifies a feature angle above which layers are collapsed automatically.

- *nSmoothSurfaceNormals* sets the number of smoothing iterations to be performed on the surface point normal.

- *nSmoothNormals* sets the number of smoothing iteration of the interior displacement field.

- *nSmoothThickness* is a number of smoothing operations that can be performed on the layer thickness.

- *maxFaceThicknessRatio* stops layer growth on highly warped cells.

- *maxThicknessToMedialRatio* and *minMedianAxisAngle* commands define the medial axis which is used when moving the mesh away from the surface; it is shown in Figure 5.10. A maximum value for the ratio of layer thickness to distance from medial axis ($\Delta H/\Delta M$) is set, above which the layer thickness is reduced.

- *nLayerIter* command sets the maximum number of layer addition iterations. If convergence is not reached it will exit the layer addition loop with the currently generated layer.

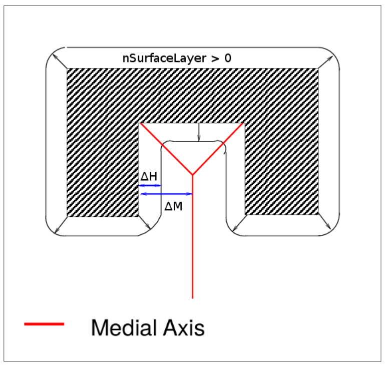
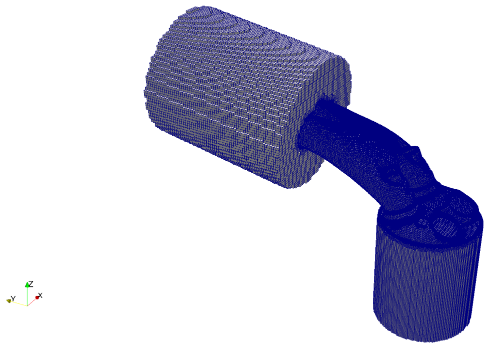


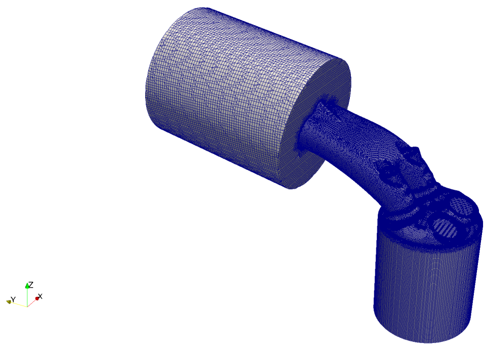
*Figure 5.10: Medial axis definition for surface layer growth*

- *nRelaxedIter* allows to achieve convergence using a set of relaxed mesh quality controls, if layer iteration has not succeeded in reaching a specific number of iterations (*nLayerIter*). With the aim of obtaining a high quality mesh, this command is commented in order to be neglected. This setup led to an increase of computing time but a better final mesh.

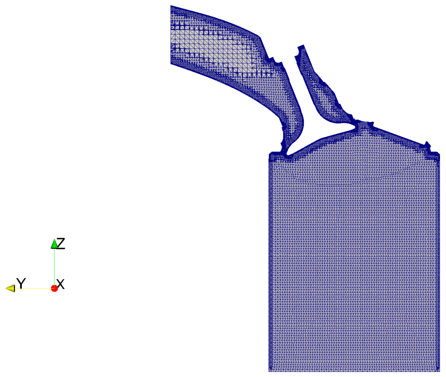
*snappyHexMesh* steps can be summarized in Figure 5.11 and the valve detail in Figure 5.12 .



(a) *snappyHexMesh* castellated step

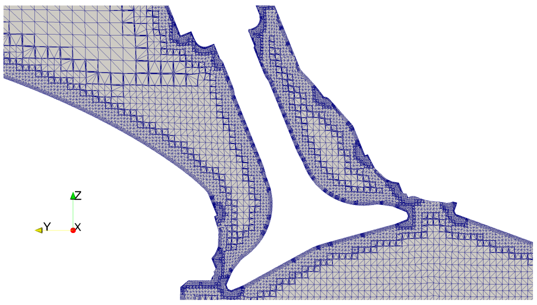


(b) *snappyHexMesh* snap step

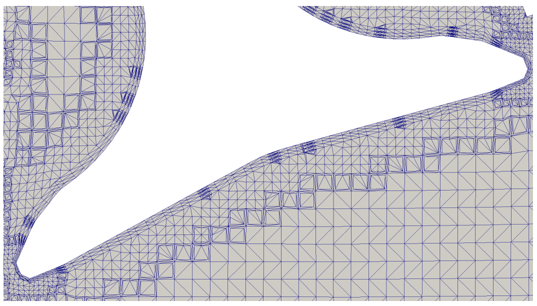


(c) *snappyHexMesh* addLayer step

*Figure 5.11: Detail valve layer zone*



(a) Valve layer 1



(b) Valve layer bottom

*Figure 5.12: Detail valve layer zone*

### 5.2.6 meshQualityControls

The last section involves *meshQualityControls*. During *snappyHexMesh* operations the mesh quality is constantly monitored. If a mesh motion or topology change introduces a poor quality cell or face, the motion or topology change is undone to revert the mesh back to a previously valid error free state.

```
meshQualityControls
{
    maxNonOrtho 60;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-30;
    minArea -1;
    minTwist 0.05;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    // Advanced
        nSmoothScale 4;
        errorReduction 0.75;
    relaxed
    {
        maxNonOrtho 75;
    }
}
```

The main control parameters of layer *meshQualityControls* are:

- face orthogonality and skewness features, which have already been discussed in section 5.1.
- *maxConcave*, shown in Figure 5.13, performs a check of the interior angles making up the face. The inserted value is the angle below which concavity is allowed.
- *minVol* and *minArea*, shown in Figure 5.14, check the minimum face area and the minimum cell pyramid volume which is calculated as:

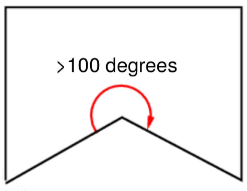


Figure 5.13: Face concavity

$$pyramidVolume = \frac{1}{3}(\mathbf{A_i} \cdot \mathbf{b_i}) \tag{5.4}$$

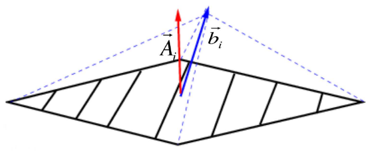


Figure 5.14: Cell pyramid volume

- $minTwist$, shown in Figure 5.15, assesses that faces are decomposed into triangular elements using the face center. The face twist is then calculated as the normalized dot product of the cell center to adjacent cell center vector with the triangular face area vector.

$$minTwist = \frac{\boldsymbol{f_i} \cdot \boldsymbol{c_i}}{|\boldsymbol{f_i}||\boldsymbol{c_i}|} \tag{5.5}$$

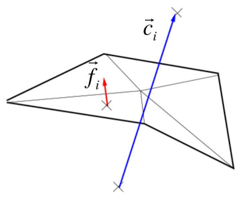


Figure 5.15: Face triangular decomposition

- *minDetermint* is calculated by taking the determinant of the tensor calculated from the face area vectors. 1 means perfect hex cell, $\leq 0$ illegal cell.

$$A = \sum_{faces} |\boldsymbol{A_i}| \tag{5.6}$$

$$t_{i,j} = \sum_{faces} \boldsymbol{A_i} x \frac{\boldsymbol{A_i}}{|\boldsymbol{A_i}|} \tag{5.7}$$

$$cellDeterminant = \left| \frac{t_{i,j}}{A} \right| \tag{5.8}$$

- *minFaceWeight* is calculated as the minimum of the projected owner cell center to face center length or neighbor cell center to face center length divided by the sum of the two lengths, Figure 5.16

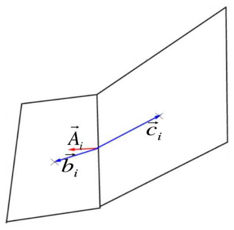


Figure 5.16: Face weight calculation

$$d_{own} = \frac{|\boldsymbol{A_i} \cdot \boldsymbol{b_i}|}{|\boldsymbol{A_i}|} \tag{5.9}$$

$$d_{neigh} = \frac{|\boldsymbol{A_i} \cdot \boldsymbol{c_i}|}{|\boldsymbol{A_i}|} \tag{5.10}$$

$$faceWeight = \frac{min(d_{own}, d_{neigh})}{d_{own} + d_{neigh}} \tag{5.11}$$

- *minVolRatio* metric is calculated as the ratio of the minimum of the owner and neighbor volume divided by the maximum of the two.

$$minVolRatio = \frac{min(V_{own}, V_{neigh})}{max(V_{own}, V_{neigh})} \tag{5.12}$$

- *nSmoothScale* applies smoothing to the displacement scaling field during each recovery iteration.
- *errorReduction* command scales back the displacement field locally where there are errors of the specified amount for each recovery iteration.

## 5.3 checkMesh

With the aim to ensure the validity of the mesh, *checkMesh* utility can be used.

```
Mesh stats
    points:           1830118
    faces:            5006408
    internal faces:   4810747
    cells:            1603949
    faces per cell:   6.12062
    boundary patches: 25
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     1397046
    prisms:        36093
    wedges:        38
    pyramids:      0
    tet wedges:    736
    tetrahedra:    4
    polyhedra:     170032
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   26658
            5   18936
            6   28242
            7   33544
            8   12812
            9   27168
           10   1044
           11   323
           12   12720
           13   104
           14   85
           15   7386
           16   1
           17   10
           18   843
           21   141
           24   15

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
             allBoundary     5020     5085  ok (non-closed singly connected)
            defaultFaces        0        0                        ok (empty)
         coronaCircolare     1914     2754  ok (non-closed singly connected)
            cylinderHead    36661    45259  ok (non-closed singly connected)
             exhaustDuct        0        0                        ok (empty)
      exhaustValveBottom     5168     5906  ok (non-closed singly connected)
    exhaustValveDiagonal        0        0                        ok (empty)
```

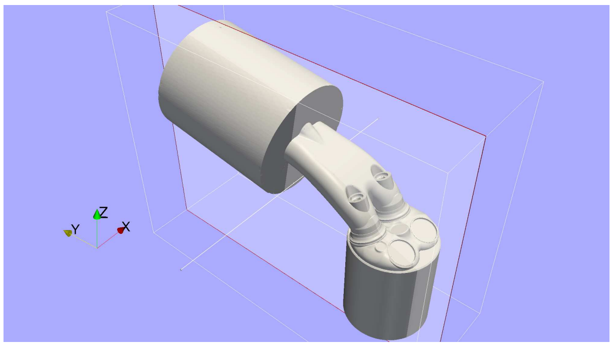
```
       exhaustValveSide     4027      5685  ok (non-closed singly connected)
       exhaustValveStem        0         0                       ok (empty)
        exhaustValveTop        0         0                       ok (empty)
              exhSeat1         0         0                       ok (empty)
              exhSeat2         0         0                       ok (empty)
                 inlet     2348      2443  ok (non-closed singly connected)
            intakeDuct    62681     74417  ok (non-closed singly connected)
           intakeSeat1     4844      6779  ok (non-closed singly connected)
           intakeSeat2    10863     14584  ok (non-closed singly connected)
     intakeValveBottom     8138      9446  ok (non-closed singly connected)
   intakeValveDiagonal     5641      8401  ok (non-closed singly connected)
       intakeValveSide     3686      5749  ok (non-closed singly connected)
       intakeValveStem     1861      2604  ok (non-closed singly connected)
        intakeValveTop     7133      9222  ok (non-closed singly connected)
                 liner    22234     23226  ok (non-closed singly connected)
                outlet         0         0                       ok (empty)
                piston         0         0                       ok (empty)
             serbatoio    13442     14966  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-0.0731569 -0.0595716 -0.11) (0.0374177 0.215324 0.109632)
    Mesh (non-empty, non-wedge) directions (1 1 1)
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (-2.1373e-16 2.83923e-15 -1.47728e-15) OK.
    Max cell openness = 4.28094e-16 OK.
    Max aspect ratio = 16.1273 OK.
    Minimum face area = 2.12099e-09. Maximum face area = 9.94223e-06.  Face area magnitudes OK.
    Min volume = 5.78369e-13. Max volume = 1.52896e-08.  Total volume = 0.00182878.
Cell volumes OK.
    Mesh non-orthogonality Max: 59.7834 average: 9.48651
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 3.41741 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```

This utility is provided by *OpenFOAM* and produces a full report with mesh quality metrics and element count statistics.

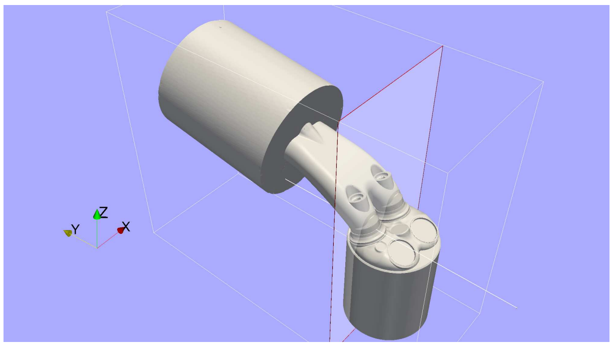
Chapter 6

# Simulation Results

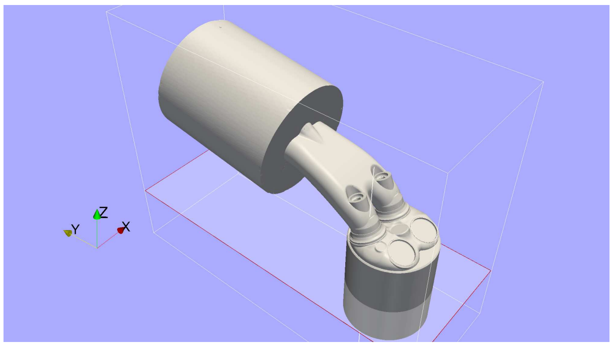
Figure 6.1 shows the section planes, on which the images that will be described later are obtained. Figure 6.1a shows the valve longitudinal section plane placed at a distance of 18 mm from the cylinder axis. Figure 6.1b shows the transverse section plane located at 24 mm. Figure 6.1c shows the cylinder section plane where the velocity maps have been calculated and it is placed at a distance of 45 mm from the cylinder top.



(a) Section plane YZ



(b) Section plane XZ



(c) Section plane XY

*Figure 6.1: Section plane*

## 6.1 Unsteady simulations

After having carried out some preliminary numerical simulations, it has been possible to observe that, starting from 5 mm valve lift, in numerical simulations the flow velocity under the intake valve is higher than experimental tests. This phenomenon may be detected for both tests but with different intensity. Several simulations are carried out in this study to investigate this flow behavior; among them, unsteady simulations have been made in order to check if unsteady phenomena could influence the velocity flow field.

For this work an unsteady solver must be used, since steady state algorithm as rhoSimpleFoam is ineffective. Among the unsteady solvers provided by OpenFOAM, rhoPimpleFoam has been used.

rhoPimpleFoam is a transient solver for compressible flow and it uses a PISO-SIMPLE merged algorithm. The SIMPLE algorithm is essentially a guess-and-correct procedure for the calculation of pressure and velocities. On the other hand, the PISO algorithm, is a pressure-velocity calculation procedure developed originally for non-iterative computation of unsteady compressible flows. For this simulations, working with a too small time step is not necessary. First, the physics of this problem does not have small timescales, then the use of low Courant number would involve an excessive duration of the simulation.

With the aim of registering unsteady phenomena, nine points have been probed in order to measure the velocity components, pressure, enthalpy, kinetic turbulent energy and specific turbulence dissipation rate at each time step. These points are placed on a plane passing through the axis of the cylinder and orthogonal to the tumble vortex rotation axis, as shown in Figure 6.2.

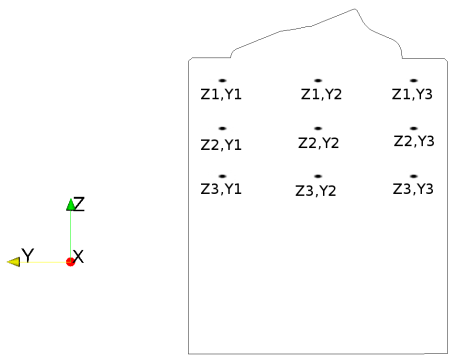


*Figure 6.2: Points distribution within the cylinder*

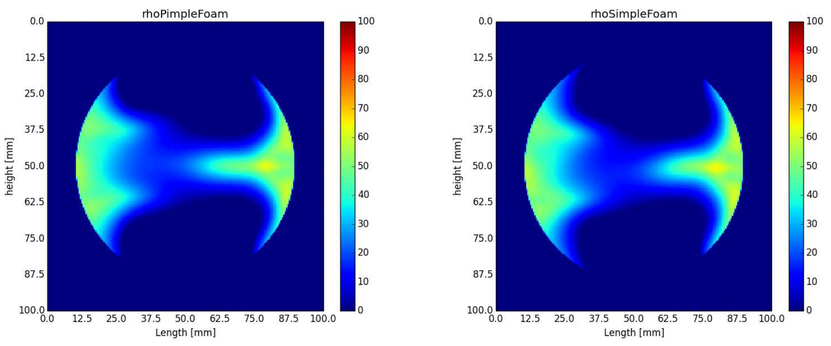
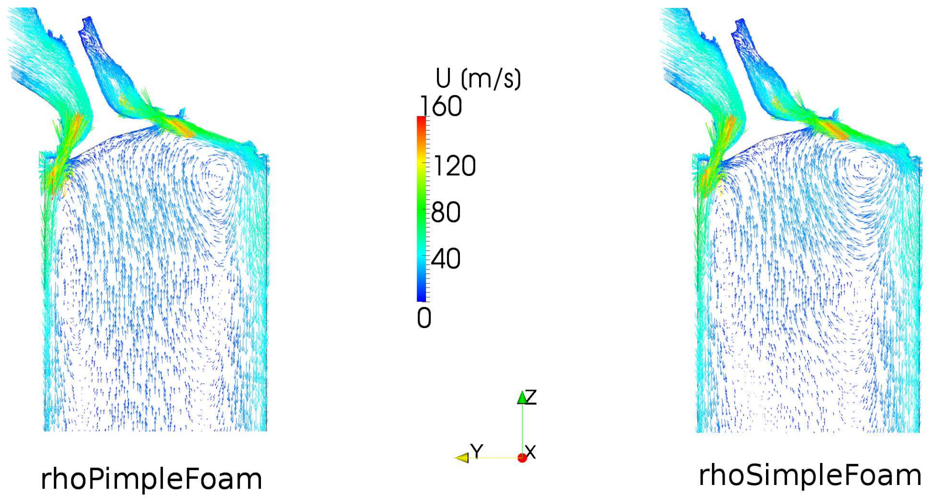
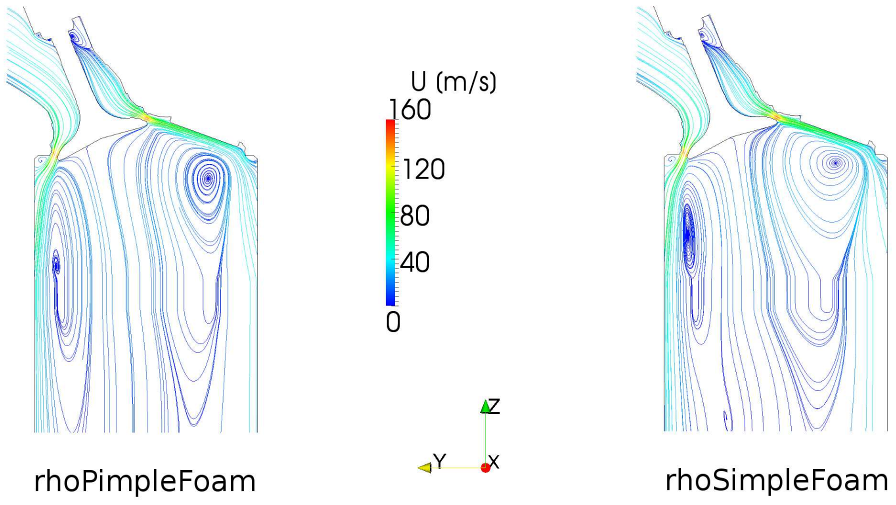


*Figure 6.3: Velocity maps rhoPimple vs rhoSimple*

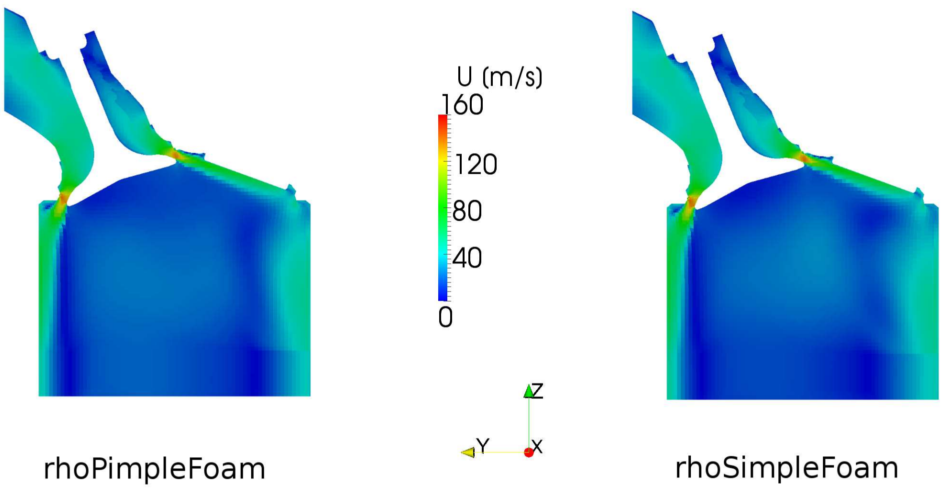


(a) Vector plot

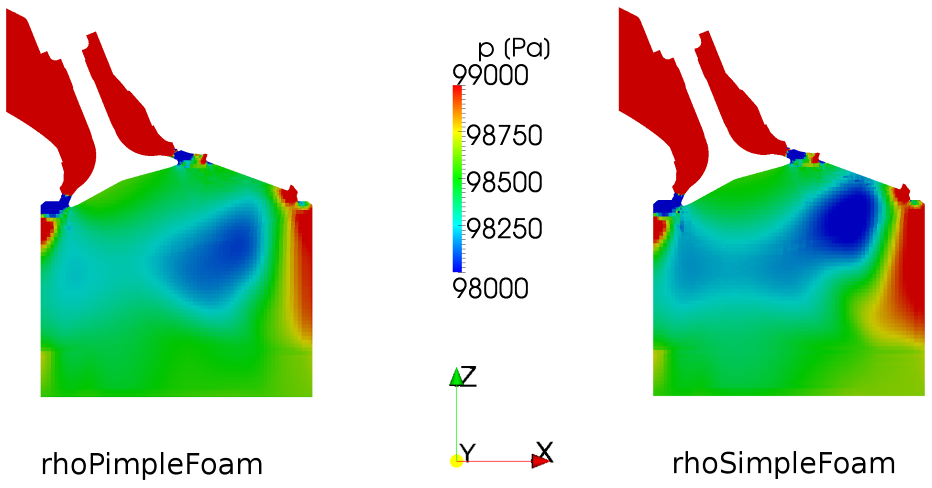


(b) Streamlines

*Figure 6.4: Vector plot and streamlines rhoPimple vs rhoSimple*

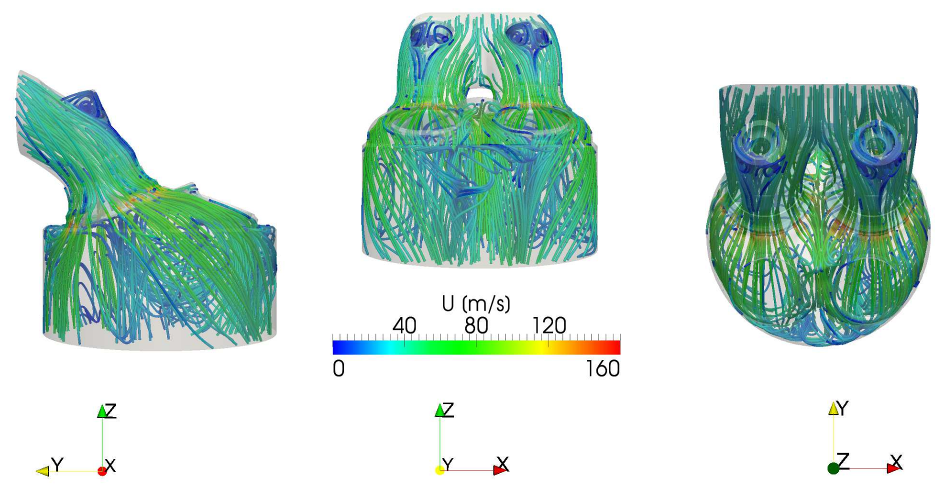


(a) Velocity field

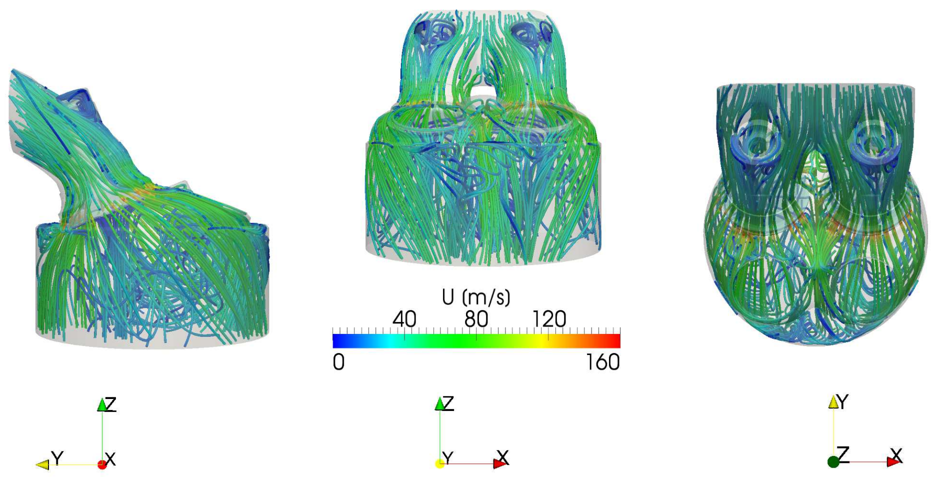


(b) static pressure contours

*Figure 6.5: Velocity and static pressure contours rhoPimple vs rhoSimple*

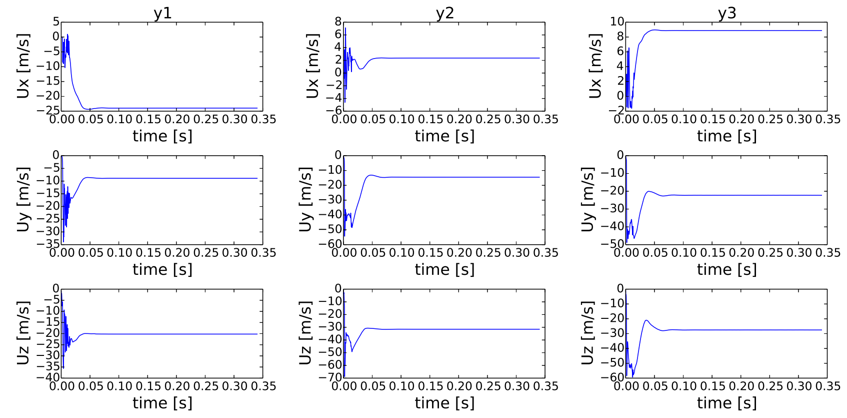


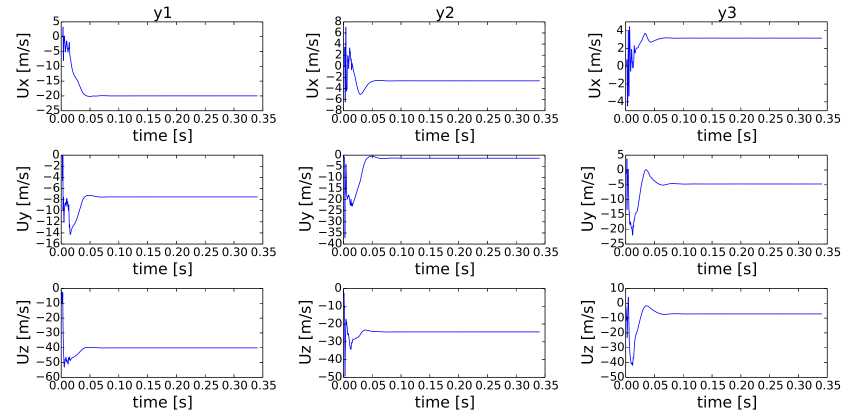
(a) Valve flow rhoPimple



(b) Valve flow rhoSimple

*Figure 6.6: Valve flow rhoPimple vs rhoSimple*


(a) Z1


(b) Z2

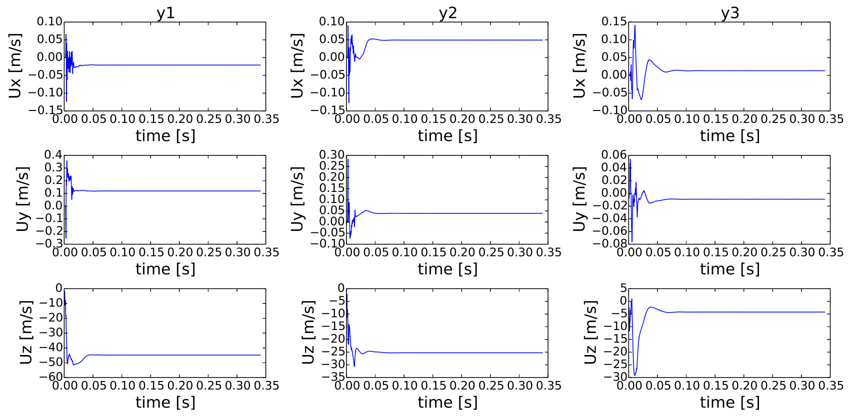
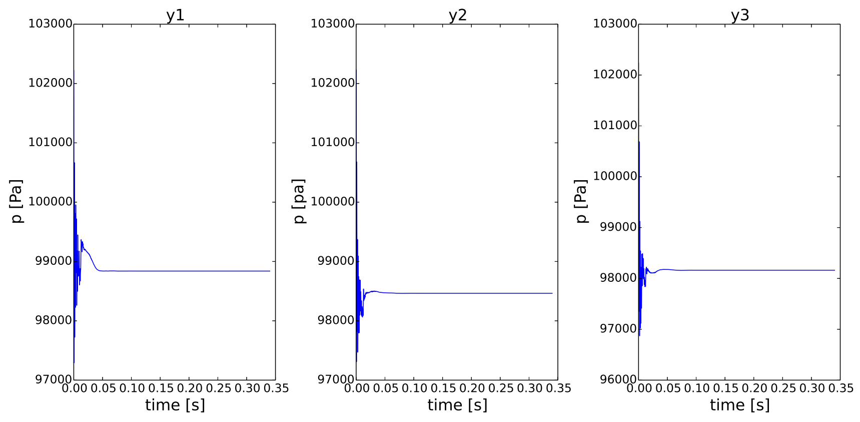
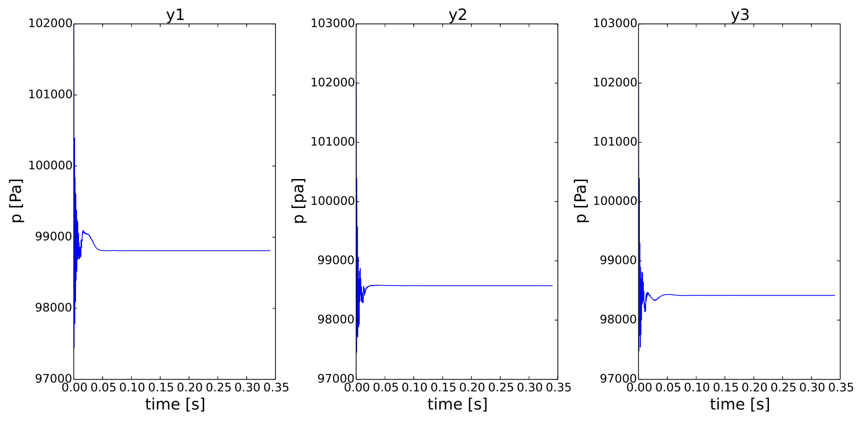

(c) Z3

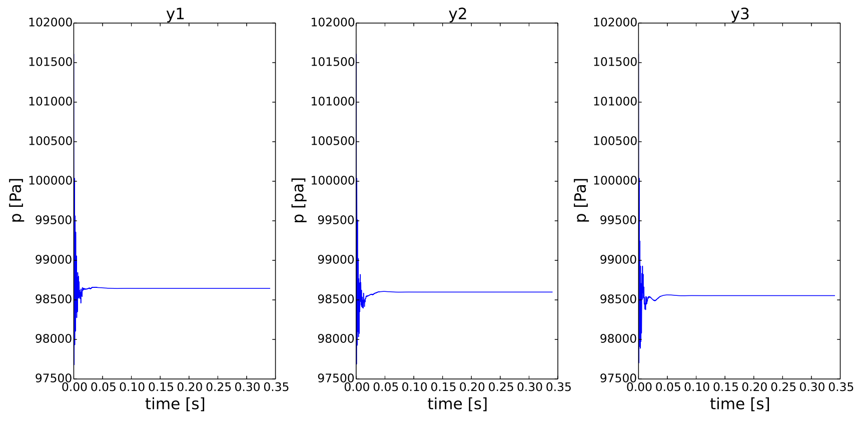
*Figure 6.7: Unsteady velocity components*



(a) Z1

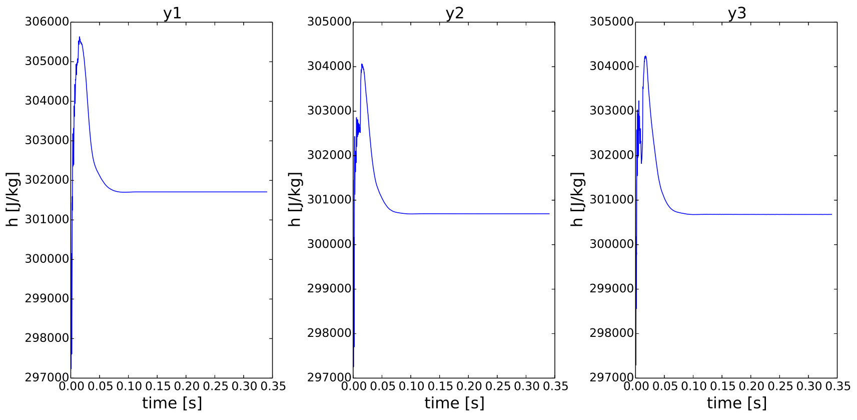


(b) Z2

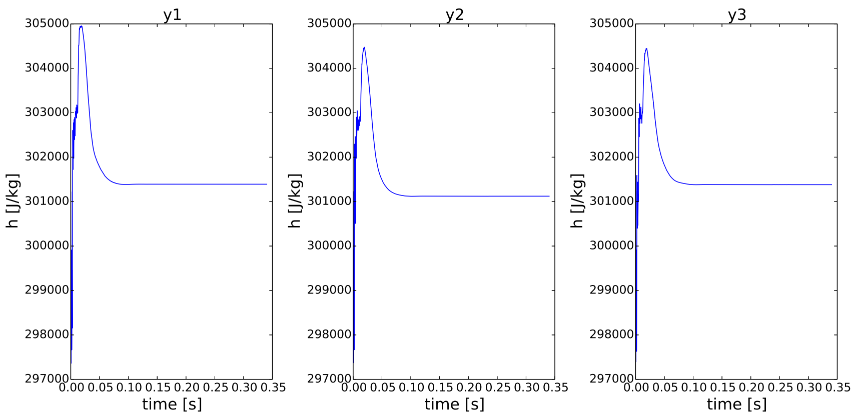


(c) Z3

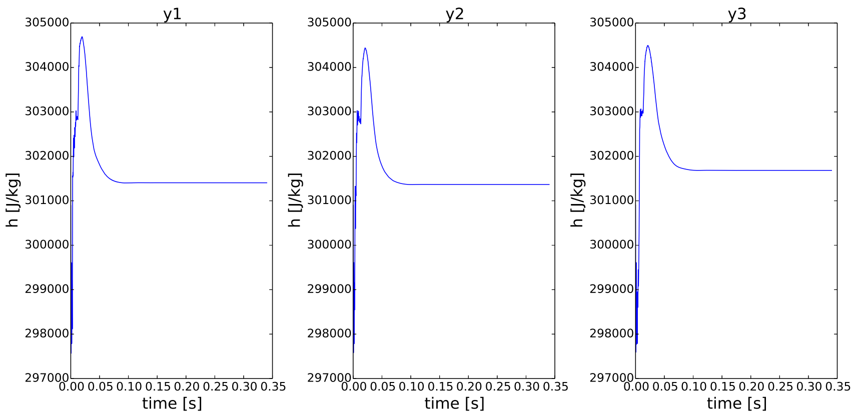
*Figure 6.8: Unsteady pressure*



(a) Z1

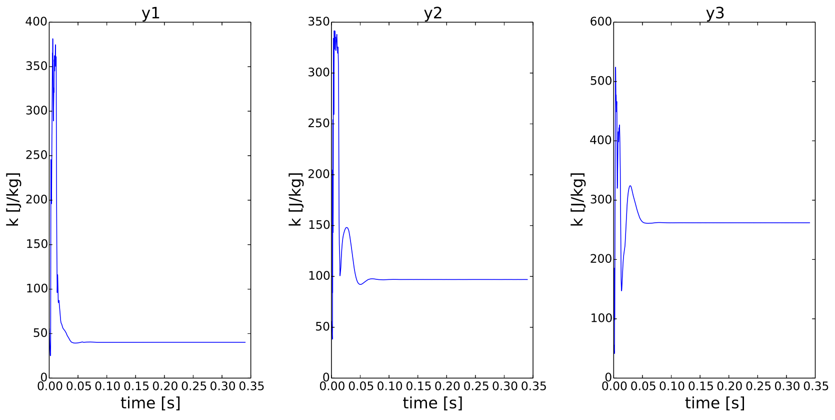


(b) Z2

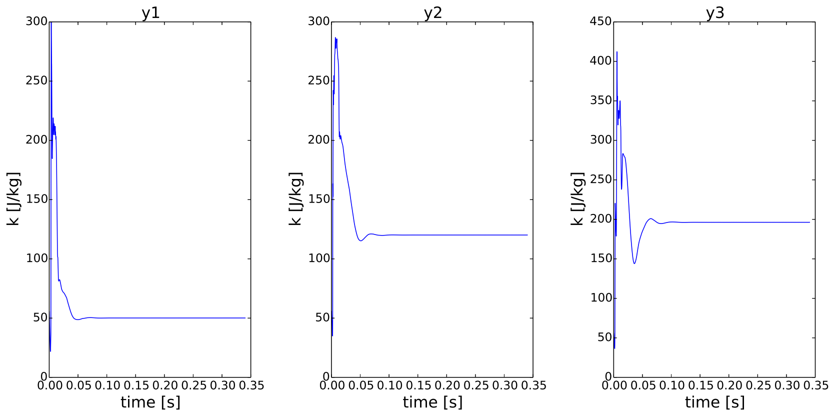


(c) Z3

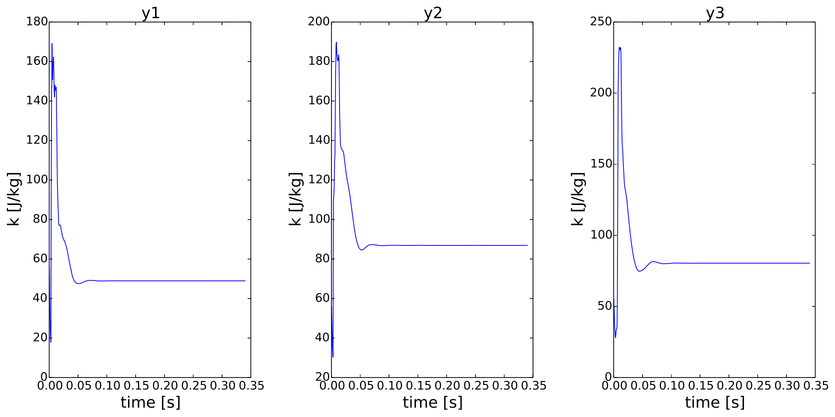
*Figure 6.9: Unsteady enthalpy*



(a) Z1

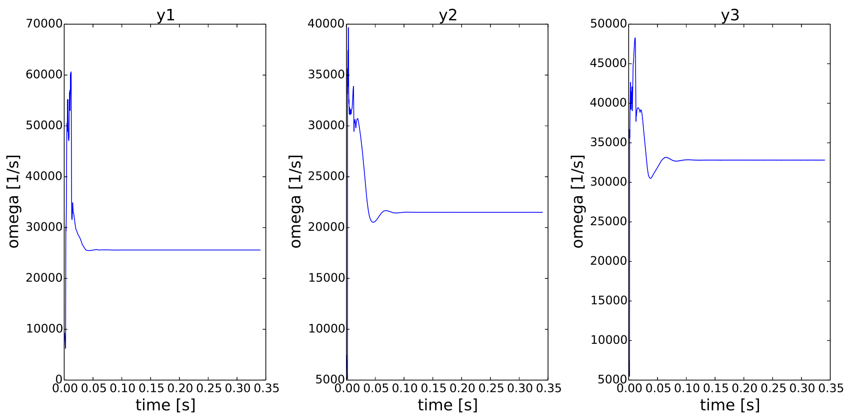


(b) Z2

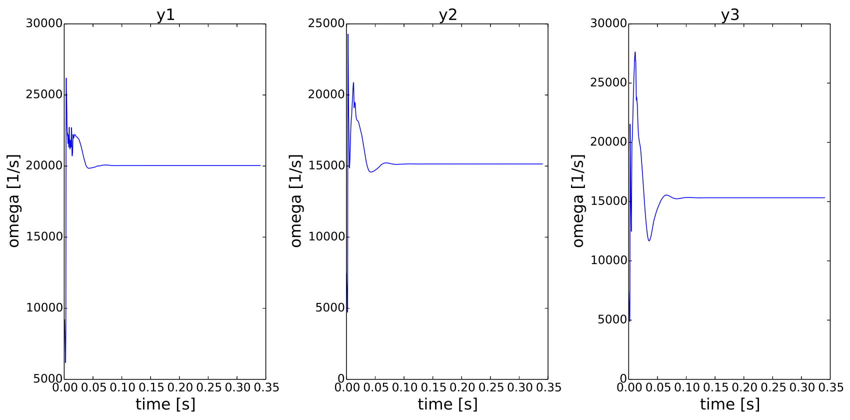


(c) Z3

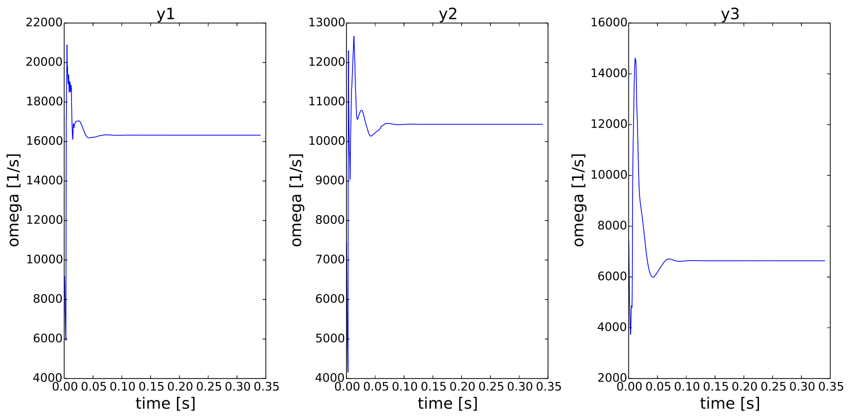
*Figure 6.10: Unsteady kinetic turbulent energy*



(a) Z1

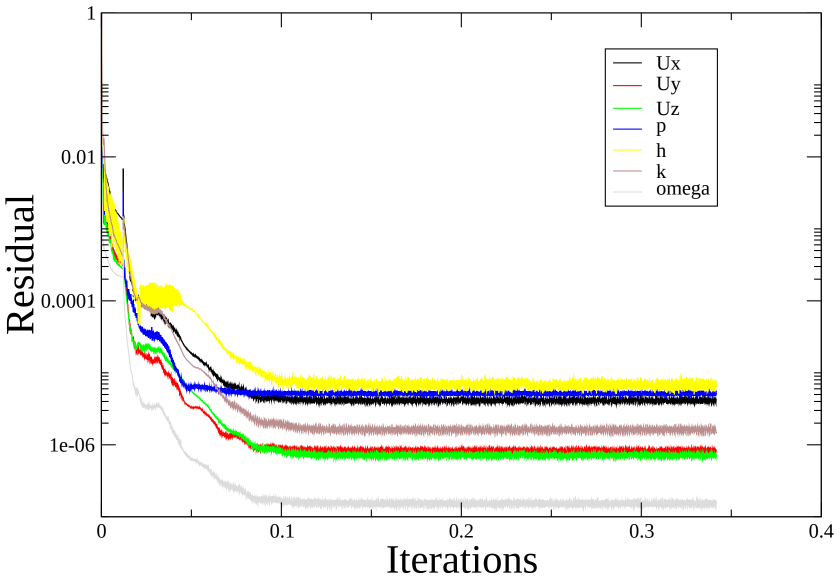


(b) Z2

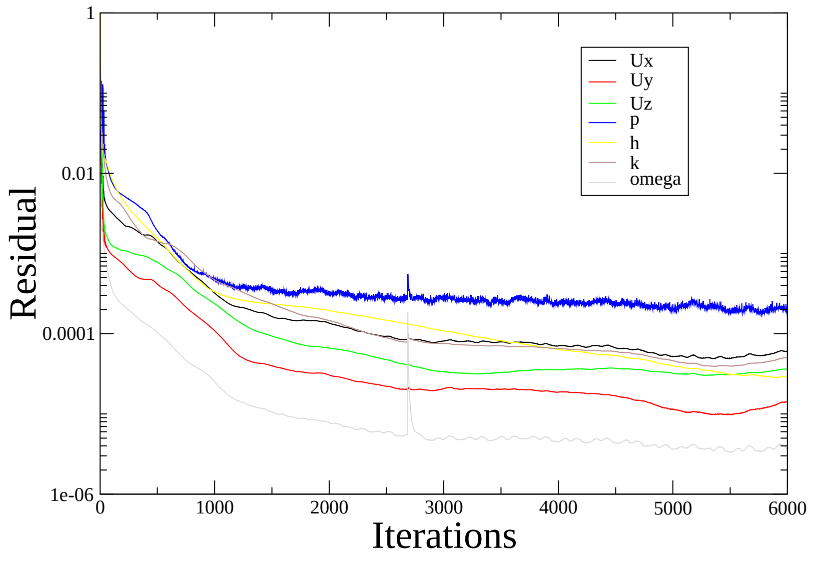


(c) Z3

*Figure 6.11: Unsteady specific turbulence dissipation rate*



(a) Unsteady residual plot



(b) Steady state residual plot

*Figure 6.12: Valve flow rhoPimple vs rhoSimple*

In Figure 6.3, Figure 6.4, Figure 6.5 and Figure 6.6 it is possible to observe the result of the comparison between unsteady and steady state simulations at 5 mm valve lift. Figure 6.12 illustrates simulation residuals plots. As can be seen, simulation reach convergence with lower values of residuals than in steady state simulation, therefore a good accuracy is expected. Figure 6.3 shows the comparison between velocity maps. It is possible to observe that there is almost no difference both in shape and values. Even in Figure 6.4 there are not significant differences: indeed streamlines and vector plots are very similar. Small difference can be noted between tumble vortexes; tumble vortex in rhoPimple looks bigger and less rounded than in rhoSimple simulation. The main difference detected is shown in Figure 6.5. Static pressure contour highlights that tumble vortex intensity in unsteady solution is less strong than in steady simulation. All these small differences could be due to simulation convergence which in unsteady simulation is slightly better.

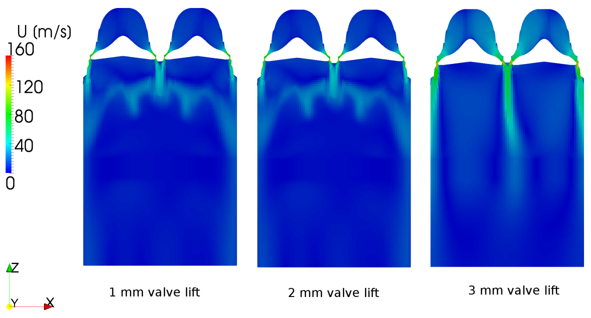
In conclusion, it is possible to state that there are not unsteady phenomena that in the flow field. In Figure 6.7 it is possible to observe that for all the three components of velocity, after a brief transitional period, the flow is characterized by steady state behavior. This feature is found also in the other quantities analyzed. Indeed Figure 6.8, Figure 6.9, Figure 6.10 and Figure 6.11 show the same behavior as velocity for pressure, enthalpy and both the turbulence quantities: kinetic turbulent energy and specific turbulence dissipation rate.

## 6.2 Steady state simulations

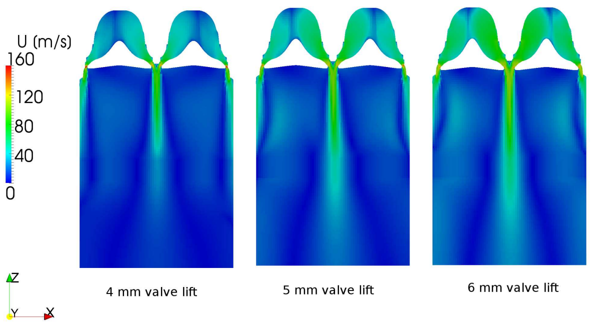
### 6.2.1 In-cylinder velocity field

In-cylinder velocity field should be symmetric. Beyond checking numerical simulation convergence through residuals, it is crucial to assess that simulation motion field makes physically sense. Since intake ducts are symmetric, the symmetry of the ingoing valve flow must be respected.

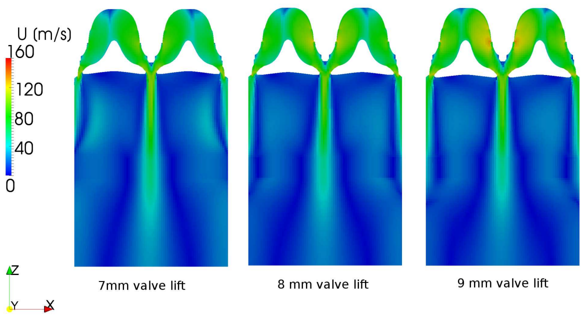
Figure 6.13 shows that symmetry is actually respected and that for low, medium and high valve lifts proper results are obtained. Only in Figure 6.13a it is possible to observe a slight asymmetry at 3 mm valve lift, but this phenomenon does not compromise the results.



(a) Low valve lift in-cylinder velocity field



(b) Medium valve lift in-cylinder velocity field



(c) High valve lift in-cylinder velocity field

*Figure 6.13: In-cylinder velocity field*

### 6.2.2 Valve flow

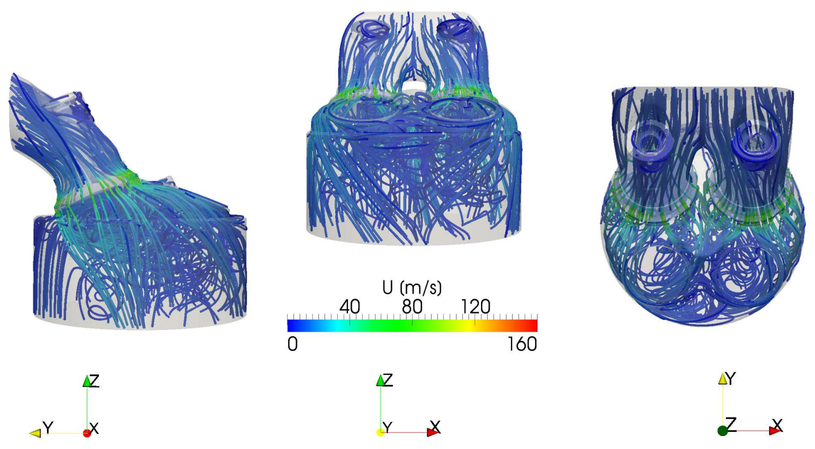
In this section outgoing valve flow is represented. To describe this phenomenon it is convenient to refer to the portion of geometry between the valve and the flow straightener.

Low valve lifts in Figure 6.14 describe a very chaotic flow behavior, especially for 1mm and 2mm valve lift, as shown in Figure 6.15a and Figure 6.15b. As previously discussed, it may be imputed to the low mass flow rate.

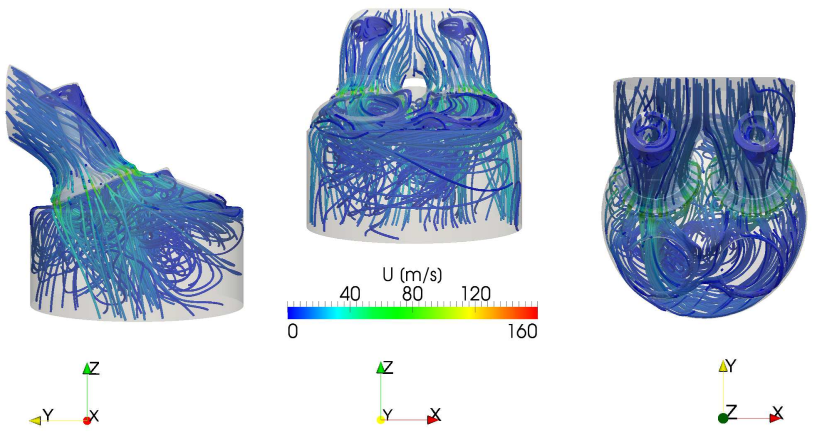
In Figure 6.15 for medium valve lifts it is possible to observe that, starting from 4mm valve lift, the flow is attached to the wall, causing less turbulence near the throat and the flow behavior has changed compared to the previous valve lifts.

High valve lifts in Figure 6.16 show that the velocity of the flow field increases proportionally to the increase of the valve lift.

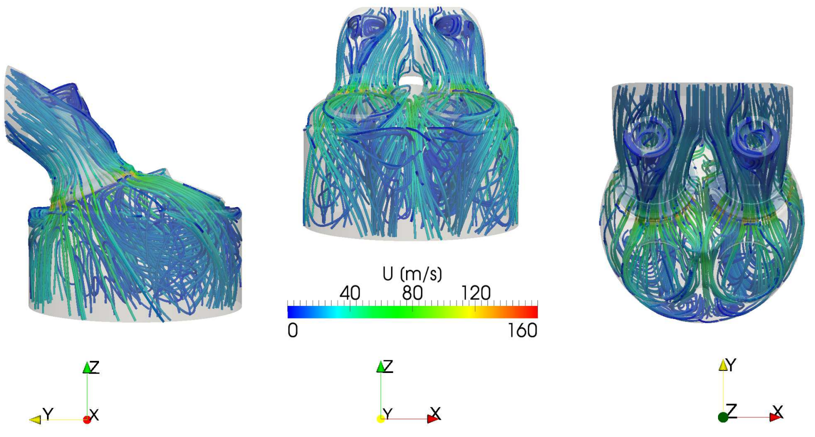
In all these figures it is possible to observe that, especially in medium and high valve lift, there is a strong upward flow towards the valve, which is due to the tumble motion. These phenomena will be discussed in the next section.



(a) 1 mm valve lift

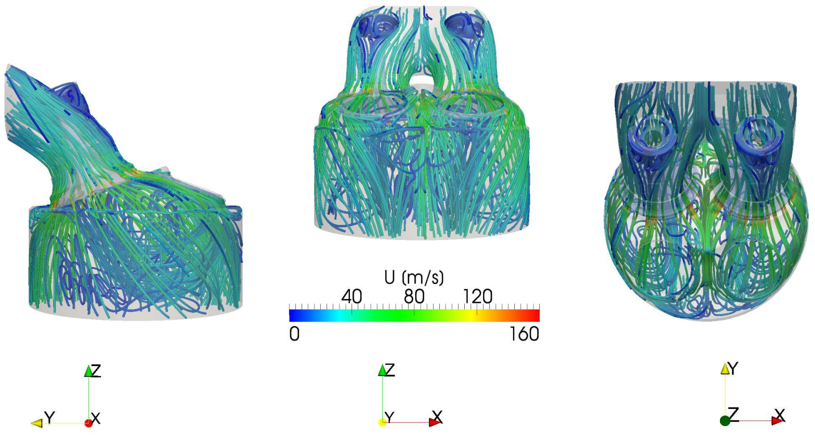


(b) 2 mm valve lift

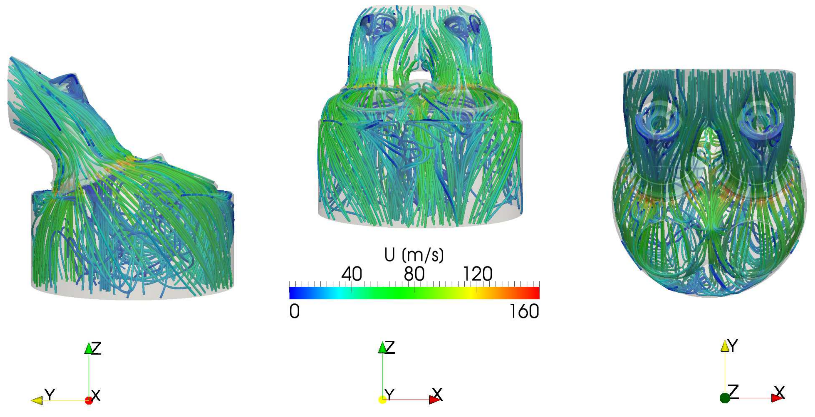


(c) 3 mm valve lift

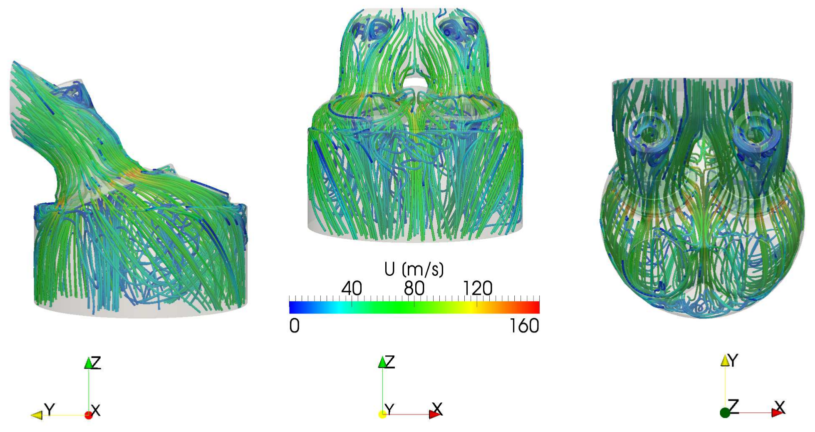
*Figure 6.14: Low valve lift valve flow*



(a) 4 mm valve lift

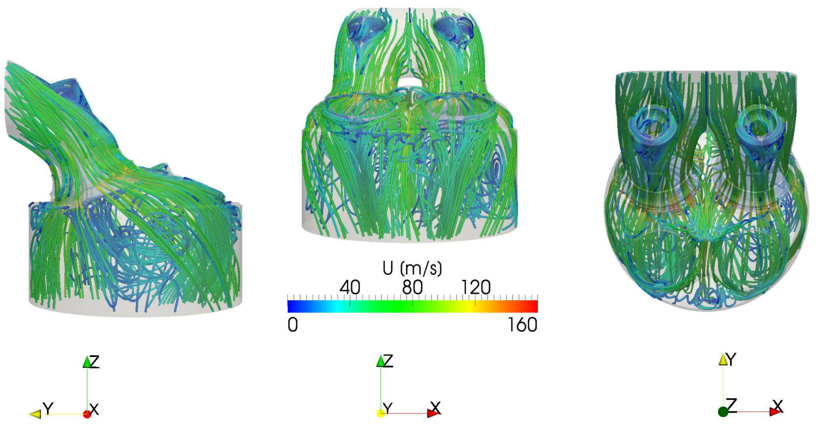


(b) 5 mm valve lift

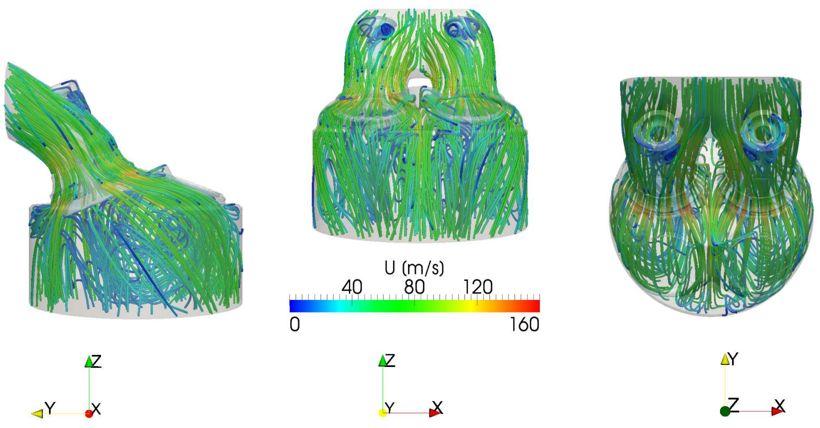


(c) 6 mm valve lift

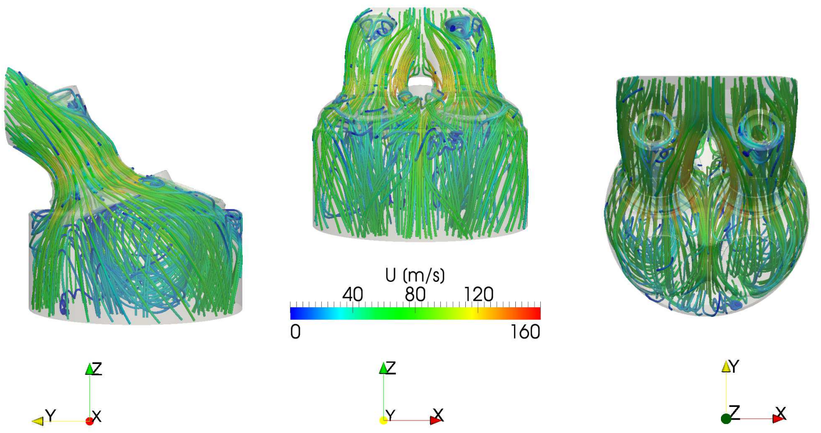
*Figure 6.15: Medium valve lift valve flow*



(a) 7 mm valve lift



(b) 8 mm valve lift



(c) 9 mm valve lift

*Figure 6.16: High valve lift valve flow*

### 6.2.3 Streamlines vs Vector plot

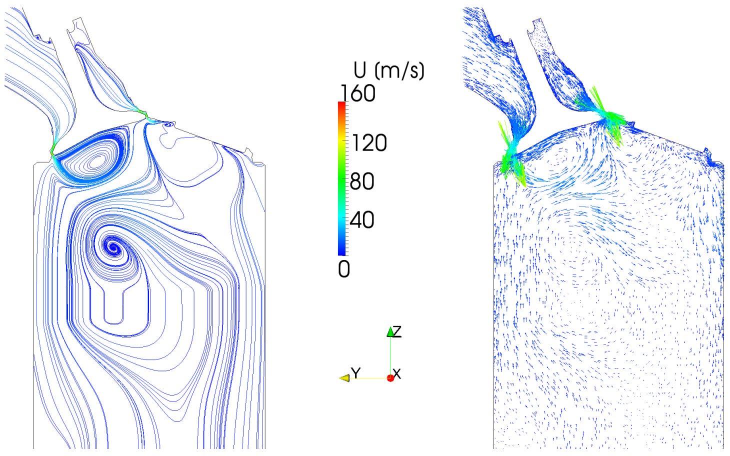
In this section streamlines and vector plot are illustrated. These results refer to the section plane shown in Figure 6.1 a.

Figure 6.17 shows streamlines and vector plot for low valve lifts. As previously explained, due to low mass flow rate entering the cylinder, the motion field is very confused. This feature is easily observable in Figure 6.17a and Figure 6.17b. Starting from 3mm valve lift, in Figure 6.17c, instead, it is possible to see the onset of two vortexes characterizing the intake stroke.

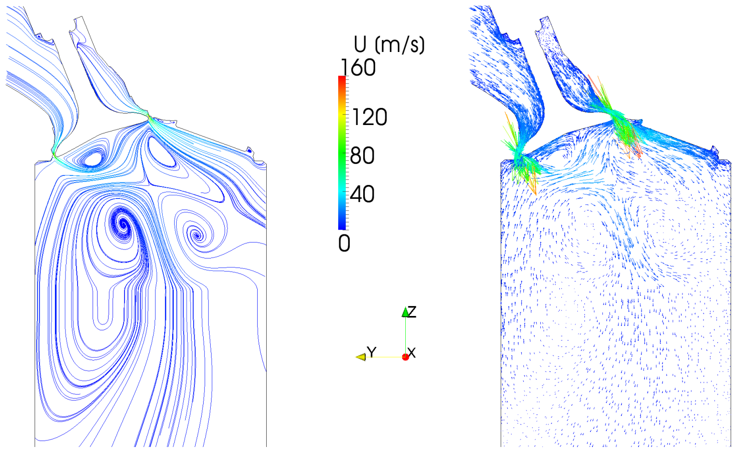
Medium valve lifts are shown in Figure 6.18, which illustrates tumble vortex development. The width of the main vortex, located under the exhaust valves, is limited by inverse tumble vortex located under the intake valves. This phenomenon is visible in Figure 6.18b and Figure 6.18c.

High valve lifts illustrated in Figure 6.19 highlight how inverse vortex dimension is reduced. After 7mm valve lift, the tumble vortex restart to grow rapidly.

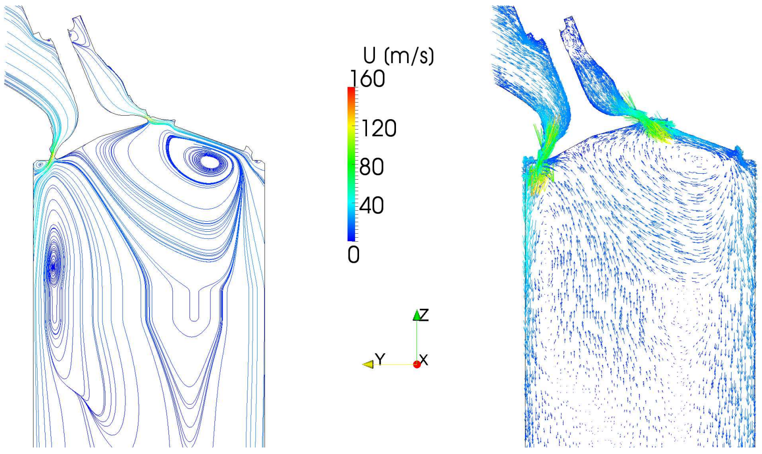
Tumble vortex development as function of valve lifts, is described in tumble coefficient section below.



(a) 1 mm valve lift

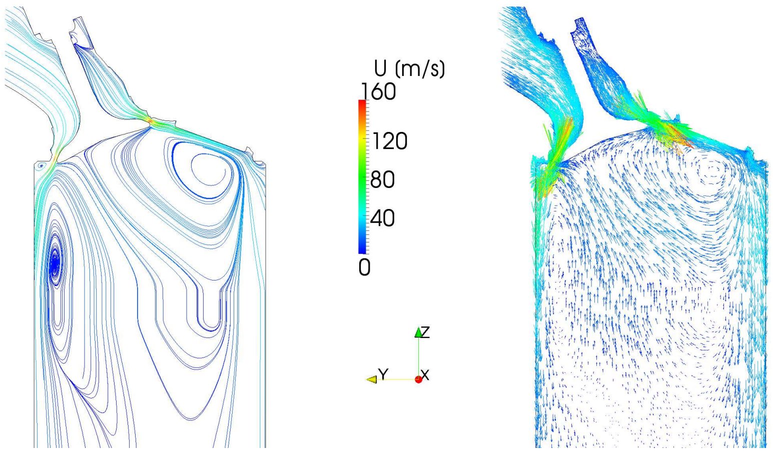


(b) 2 mm valve lift

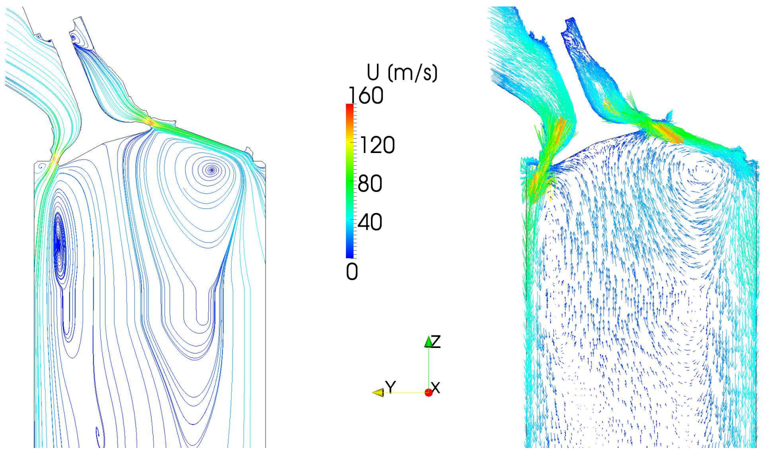


(c) 3 mm valve lift

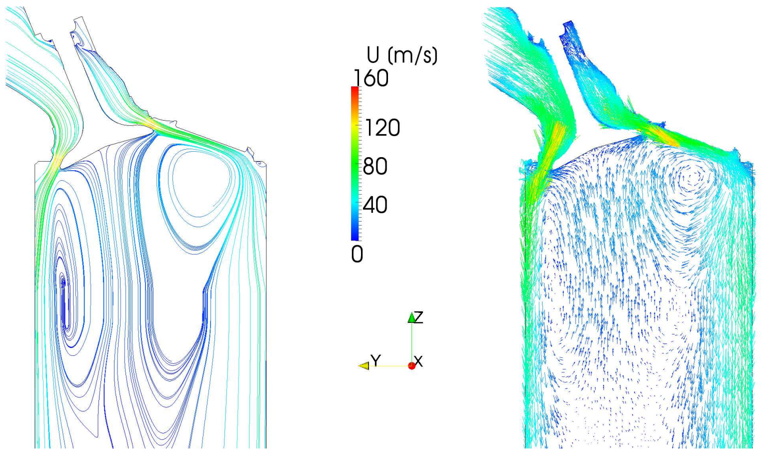
*Figure 6.17: Low valve lift streamlines vs vector plot*



(a) 4 mm valve lift

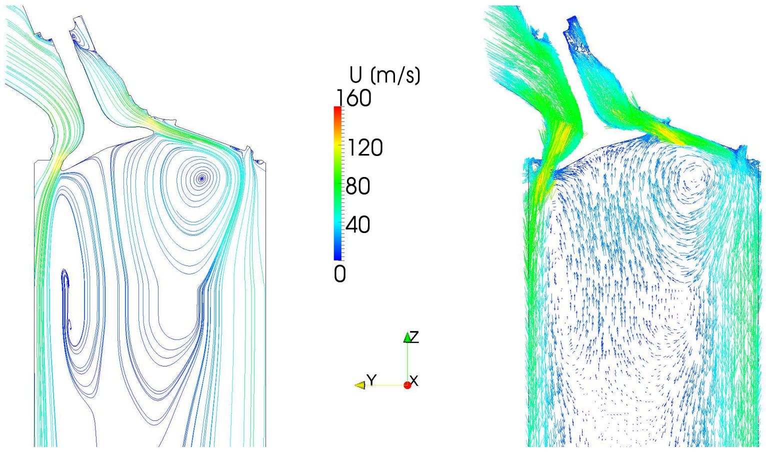


(b) 5 mm valve lift

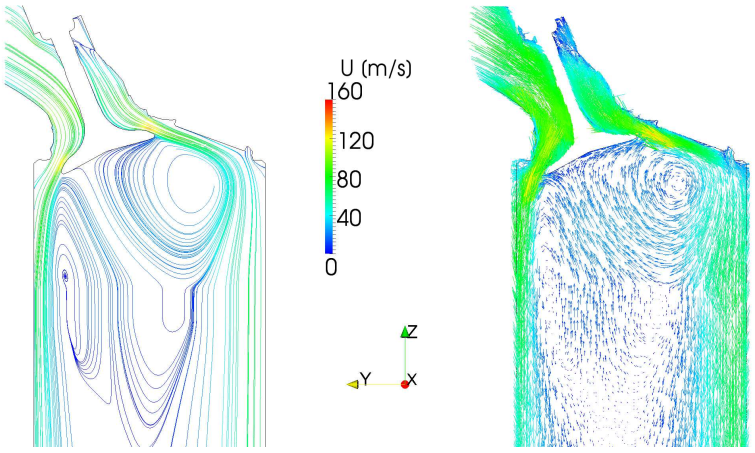


(c) 6 mm valve lift

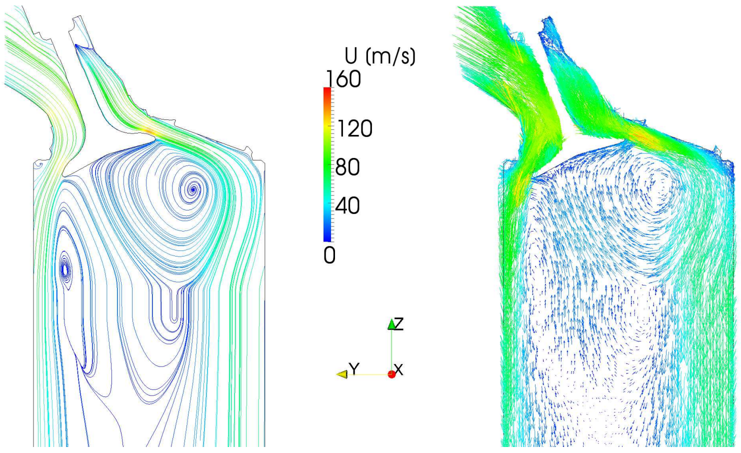
*Figure 6.18: Medium valve lift streamlines vs vector plot*



(a) 7 mm valve lift



(b) 8 mm valve lift



(c) 9 mm valve lift

*Figure 6.19: High valve lift streamlines vs vector plot*

### 6.2.4 Velocity field vs Static pressure contours

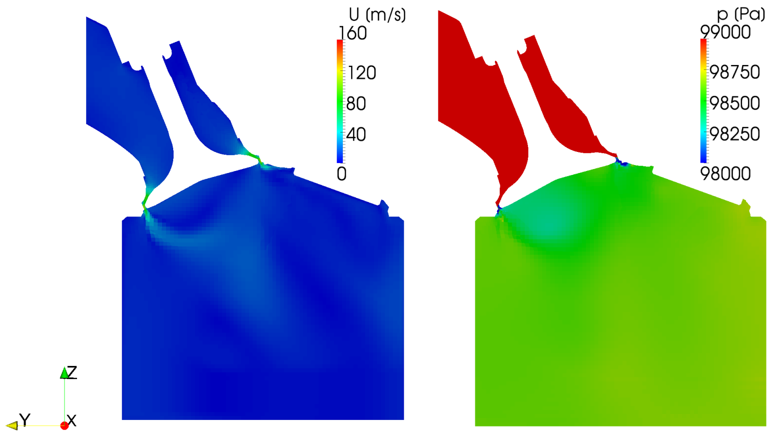
In this section velocity field and static pressure contours are illustrated.

A strong pressure differential exists near the throat which results in flow acceleration through the throat and hence high velocity. At lower lifts, shown in Figure 6.20, the jet of flow coming out of the throat causes recirculation below the valve head. A portion of high velocity flow out of the throat creates a stronger tumble inside the engine cylinder at high lifts compared to low lifts.

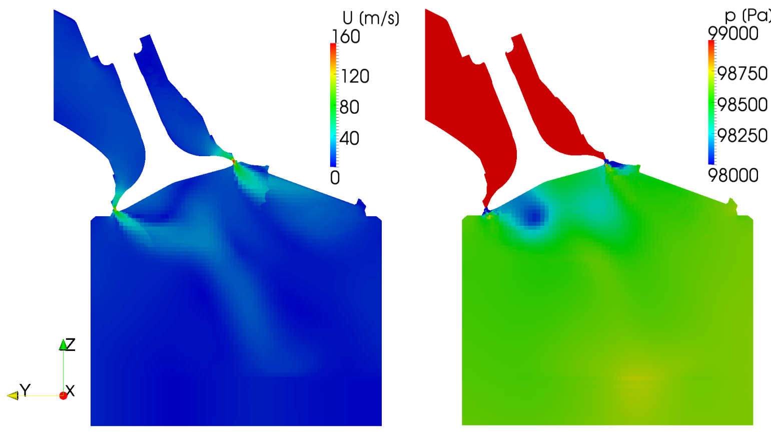
The pressure field describes the intensity of tumble vortex. Starting from low lifts it is possible to observe tumble vortex intensity growing.

As described in the previous section, at medium lifts the tumble vortex does not increase its size, but in figure Figure 6.21 it is shown that its intensity has grown.

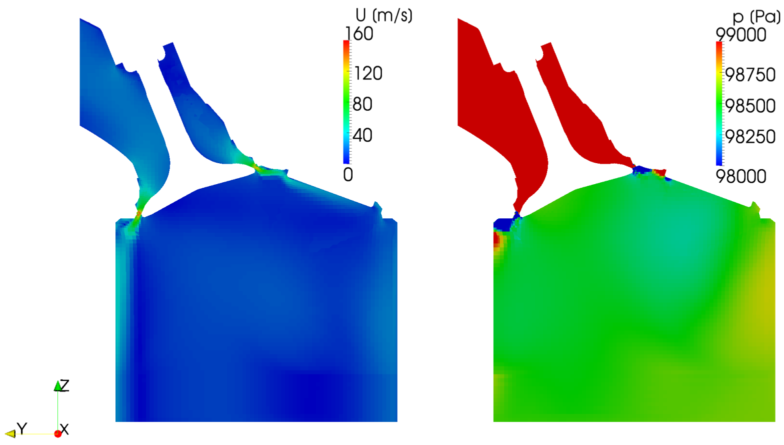
Figure 6.22 shows that at higher valve lifts the tumble vortex is bigger than at previous lifts and it moves from the zone below the exhaust valve to the zone below the intake valve, pushing the inverse vortex against the wall.



(a) 1 mm valve lift

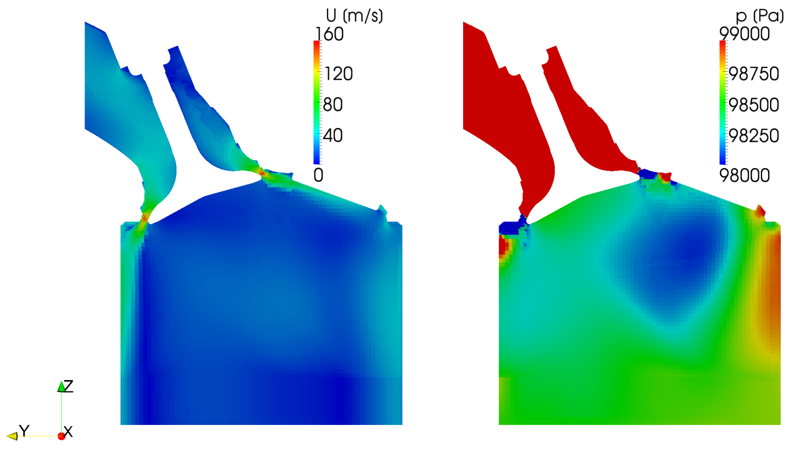


(b) 2 mm valve lift

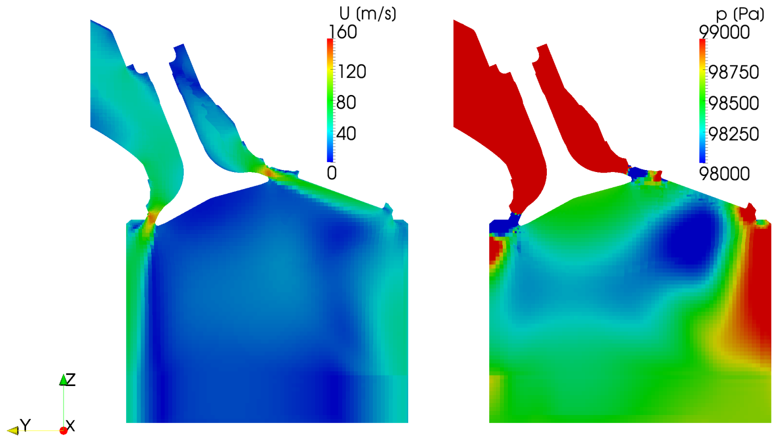


(c) 3 mm valve lift

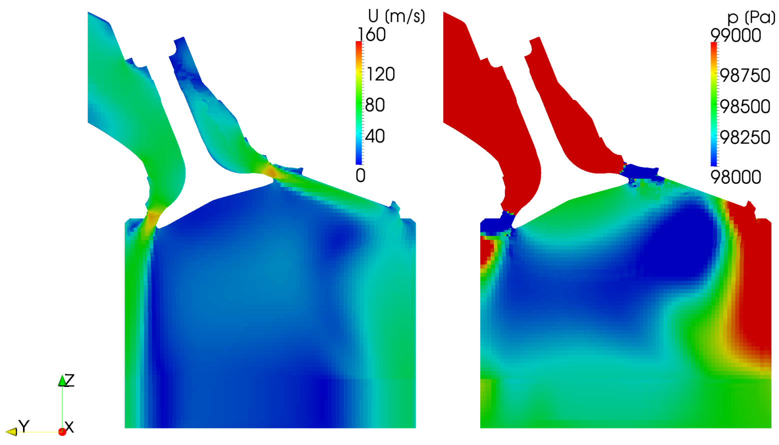
*Figure 6.20: Low valve lift velocity field vs static pressure contours*



(a) 4 mm valve lift

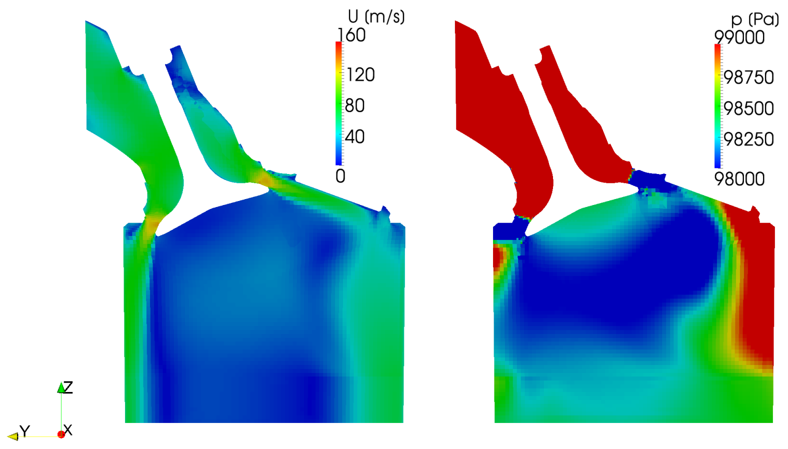


(b) 5 mm valve lift

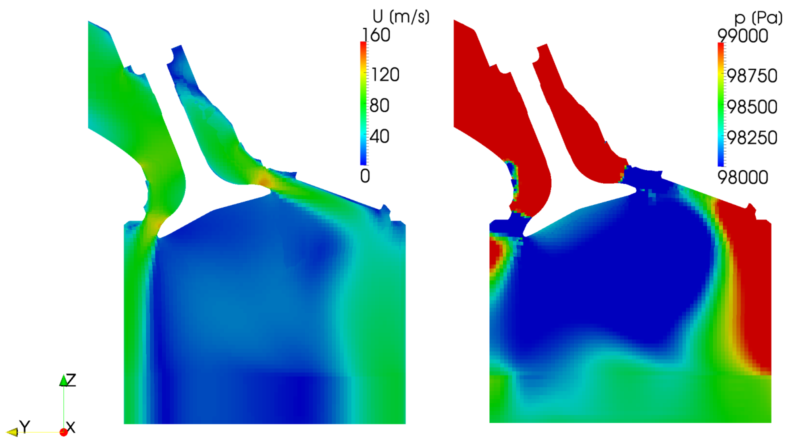


(c) 6 mm valve lift

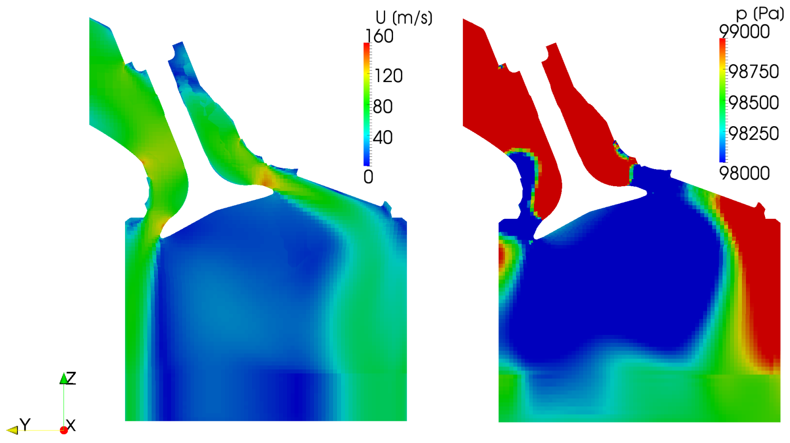
*Figure 6.21: Medium valve lift velocity field vs static pressure contours*



(a) 7 mm valve lift



(b) 8 mm valve lift



(c) 9 mm valve lift

*Figure 6.22: High valve lift velocity field vs static pressure contours*

### 6.2.5 Velocity maps

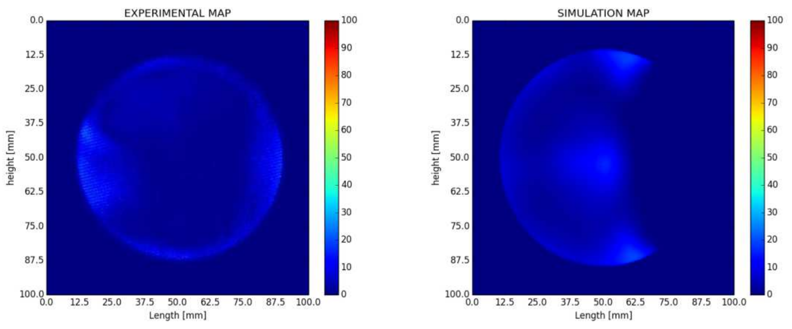
Velocity maps represent the velocity component of each cell that is parallel to the cylinder axis. These velocities have been acquired thanks to *sample* utility provided by *OpenFOAM*. Subsequently they have been elaborated through python script and organized in a velocity matrix, whose size is 200 x 200.

The left side of velocity maps represents the zone below the exhaust valve. The opposite side, instead, represents the zone below the intake valve.

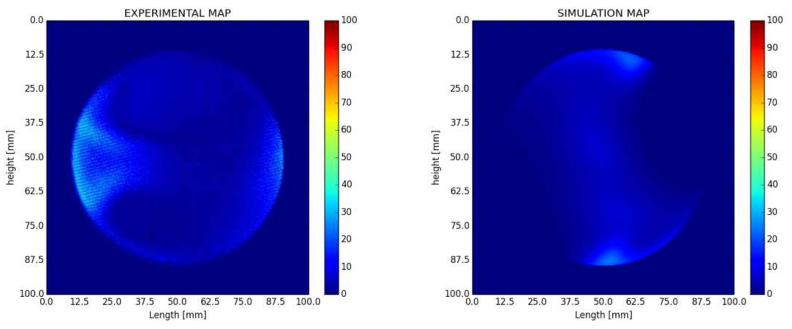
In both the experimental tests and the numerical simulations the velocity field starts to be actually symmetric to the horizontal axis of the section from 5mm valve lift included. It is possible to observe the classic air distribution below the exhaust valve, that represents a good direct tumble. There is an air zone under the intake valve that contributes to generate inverse tumble which reduces the direct tumble effect, as it is shown in streamlines and vector plot figures.

In experimental tests this zone begins to be visible at 4mm valve lift, increases its intensity and dimension until 7mm valve lift and decreases dimension and intensity at 8mm and 9mm valve lift, thus favoring a more intense tumble. In the simulations, instead, the zone under the intake valve decreases dimension and intensity in a less considerable way compared to experimental tests: indeed, this phenomenon occurs only at 9mm valve lift.

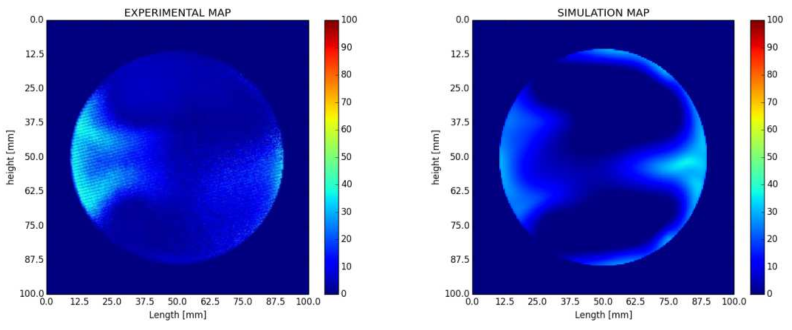
The valve lift lower than 5mm doesn't present symmetry to the horizontal axis. FIAT claims that these conditions are due to imperfect mechanical manufacturing. Actually, as it is possible to observe in streamlines figures, they are a consequence of two elements: the distance between the probe and the honeycomb flow straightener as well as the low mass flow rate. For 1mm and 2mm valve lift it is not possible to make any other consideration because of the low air velocity.



(a) 1 mm valve lift

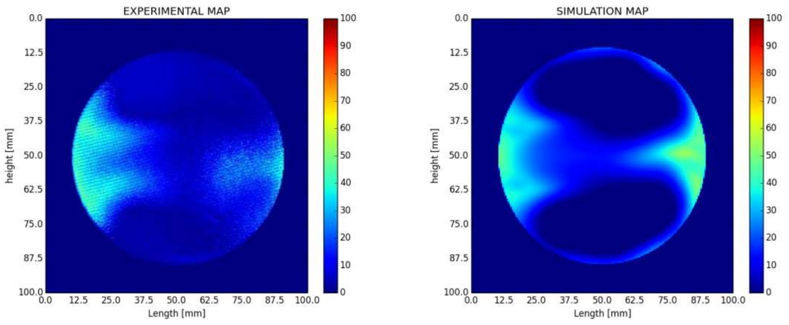


(b) 2 mm valve lift

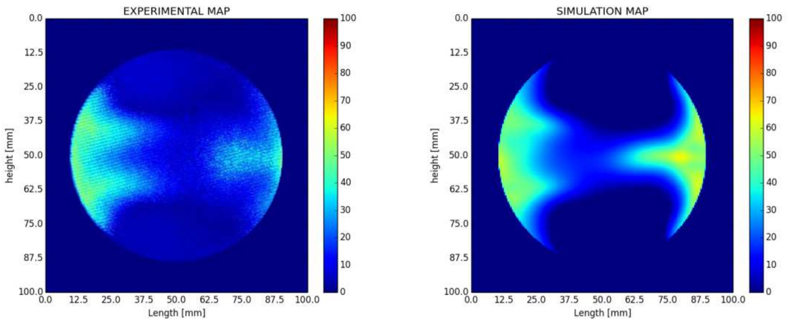


(c) 3 mm valve lift

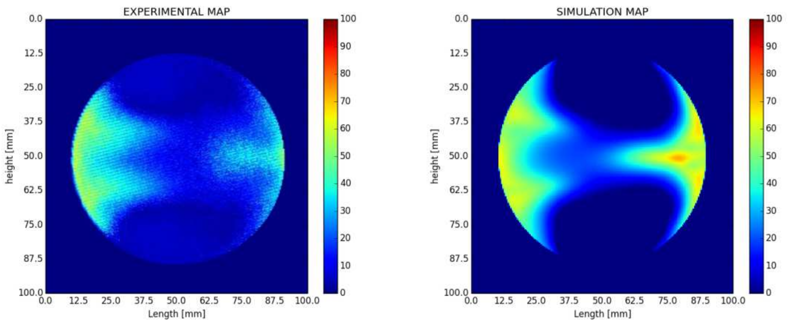
*Figure 6.23: Low valve lift velocity maps*



(a) 4 mm valve lift

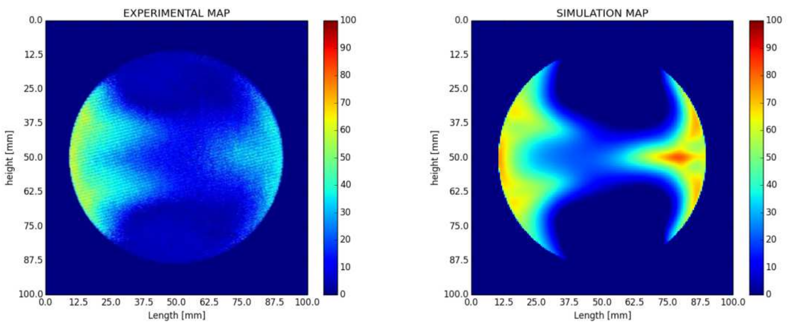


(b) 5 mm valve lift

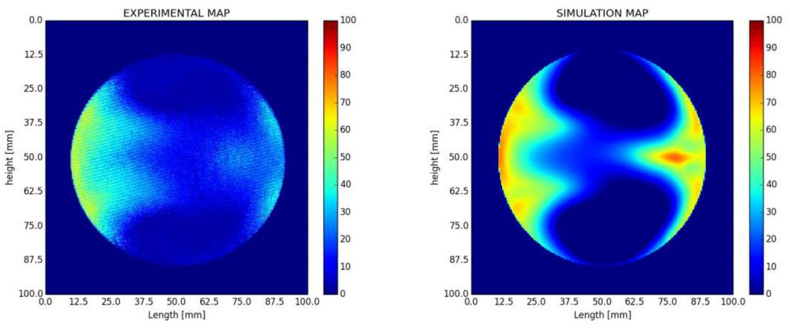


(c) 6 mm valve lift

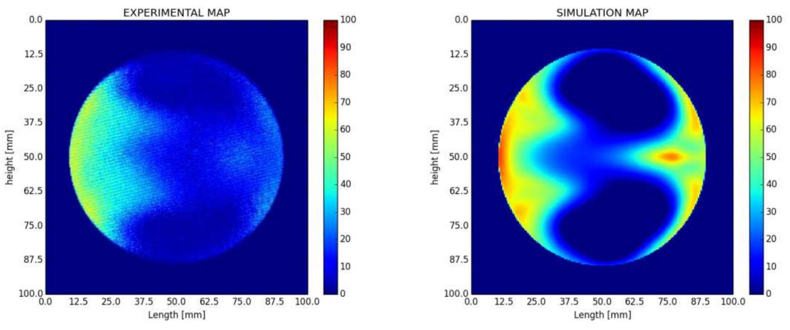
*Figure 6.24: Medium valve lift velocity maps*



(a) 7 mm valve lift



(b) 8 mm valve lift



(c) 9 mm valve lift

*Figure 6.25: High valve lift velocity maps*

### 6.2.6 Discharge coefficient

The discharge coefficient $C_D$ is defined as the ratio between the measured flow rate at standard conditions and a reference mass flow rate [7].

$$C_D = \frac{\dot{m}_a}{\dot{m}_{is}} \tag{6.1}$$

The discharge coefficient is always less than 1 because ideal conditions are never achieved for the following reasons:

- the fluid has real behavior;
- the velocity field in the section is nonuniform;
- heat transfer and viscous dissipation;
- the difference between the actual minimal area and the reference one.

The reference area is represented by the cross section of valve seat, which is constant. The discharge coefficient can be rewritten as:

$$C_D = \frac{\dot{m}_a}{A_{ref}\dot{v}_{is,spec}\rho_{01}} \tag{6.2}$$

where $\dot{v}_{is,spec}$ is defined as:

$$\dot{v}_{is,spec} = \Phi_c \sqrt{2\frac{(P_{01} - P_2)}{\rho_{01}}} \tag{6.3}$$

where the compressibility factor $\Phi_c$ is defined as:

$$\Phi_c = \sqrt{\frac{kP_{01}}{2(P_{01} - P_2)}} \sqrt{\frac{2}{k-1}\left(\frac{P_2}{P_{01}}\right)^{2/k}\left[1 - \left(\frac{P_2}{P_{01}}\right)^{\frac{k-1}{k}}\right]} \tag{6.4}$$

The discharge coefficient depends on:

- the system geometry;
- the characteristics of the fluid;
- the velocity field.

The last point can be summarized through Reynolds and Mach non-dimensional number. For low Mach number (Mach $\leq 0.7$), the discharge coefficient is only function of Reynolds number and becomes a constant for highly turbulent flow. For very turbulent regimes, as in our case, the

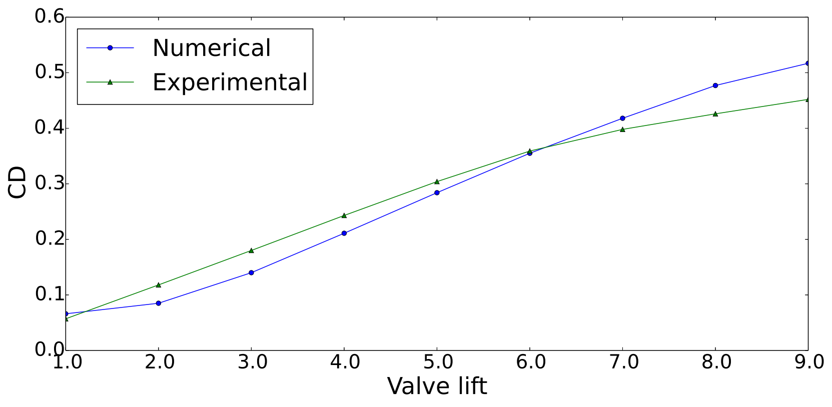


*Figure 6.26: Discharge coefficient*

discharge coefficient is independent from Reynolds number and hence from the pressure ratio across the valve that is the difference of the pressure over the valve.

Numerical discharge coefficient matches closely with experimental discharge coefficient at lower lifts and starts to be over-predicted at higher valve lifts. This could be because at higher lifts the flow starts to separate from the wall creating turbulence near the throat. The flow thus rushing into the chamber creates more intake upstream of the throat which results in more air flow at high lift position of the valve.

### 6.2.7 Tumble number

An organized motion that is used in the most modern spark ignition engines is the tumble motion [8]. The velocity maps represent the velocity field at the output section of the cylinder. These maps show an area of high velocity below exhaust valve, basic condition for tumble. To quantify the tumble, it is assumed that the vortex closes on itself on the floor where velocities are measured. The tumble vortex is developed around an axis that is orthogonal to the cylinder axis. It is possible to assume that the rotation around this axis occurs as rigid body motion. The tumble number is calculated as:

$$N_T = \frac{\omega_{average} D}{v_{is}} \tag{6.5}$$

The problem lies in the calculation of the average angular velocity, so an equivalent quantity is used. The first step consists in dividing the map into two parts on the vortex rotation axis. The velocity that can be found on the left side, creating a vortex outgoing from the left side and entering in the right side, represents the direct tumble vortex. Referring to velocity maps the velocity under intake valve generates the tumble inverse vortex. Each velocity has a certain distance $b_i$ from the axis of rotation. The angular velocity for each point can be calculated as:

$$\omega_i = \frac{v_i}{b_i} \tag{6.6}$$

The second step is to calculate an average value for the angular velocity of left and right side of the map:

$$\omega_{sx-dx} = \frac{\sum v_i b_i}{\sum b_i^2} \tag{6.7}$$

The average angular velocity to place in equation (**??**) is then:

$$\omega_{average} = \omega_{sx} - \omega_{dx} \tag{6.8}$$

As shown in vector plot in Figure 6.18b, the tumble motion is developed around an axis that is orthogonal to the cylinder axis, and it is due to the interaction between the air flow deriving from the inlet valve and the opposite cylinder wall. It is traditionally used to increase the turbulence level in combustion stroke of spark-ignition engine. Generally the ability to produce organized motion of the inlet flow corresponds to an increase in fluid-dynamic losses and consequently results in a decrease of discharge coefficient. This phenomenon can be observed from 7mm valve lift; the tumble number in simulations increases less than in experimental tests but the discharge coefficient is higher.

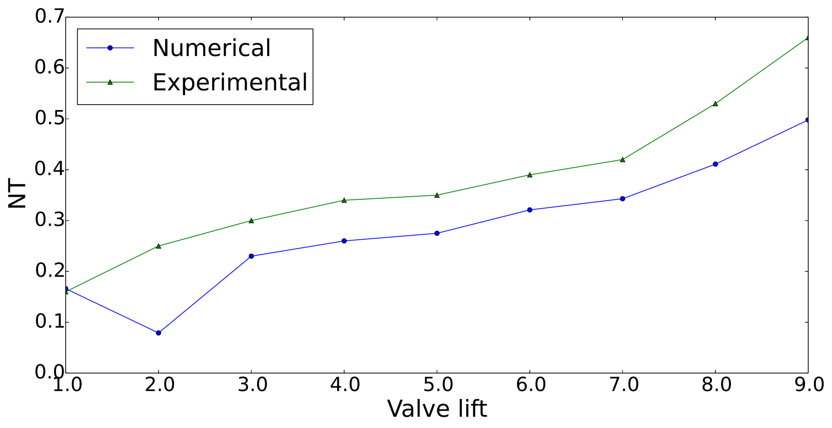


*Figure 6.27: Tumble number*

Analyzing Figure 6.27 and the vector plot figures it is possible to observe as the tumble vortex is powered by the increasing air entering through the valve. The offset between the two lines could be the result of the air velocity below the intake valve of the numerical simulations, that is higher than experimental tests. This implies a reduction of the main tumble vortex, caused by the inverse tumble vortex.

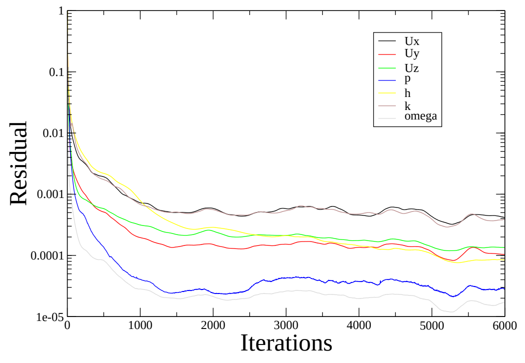
### 6.2.8 Convergence results

Figure 6.28 shows low valve lifts (1 mm, 2 mm, 3 mm) residuals plots. Low valve lifts have been the most difficult to execute. Simulations had stability problems at early iterations, for this reason robust numerical schemes have been used. 1 mm valve lift presents higher final residuals than all other simulations and they have an oscillatory trend. 2 mm valve lift has been the most problematic to simulate. The residual plot shown in Figure 6.28 b points out a cusp in the neighborhood of the 3000*th* iteration. This trend could be due to an imperfect boundary condition of $k$ or $\omega$. 3 mm valve lift is more valid than previous simulations, since residuals reach convergence before the 6000*th* iteration.

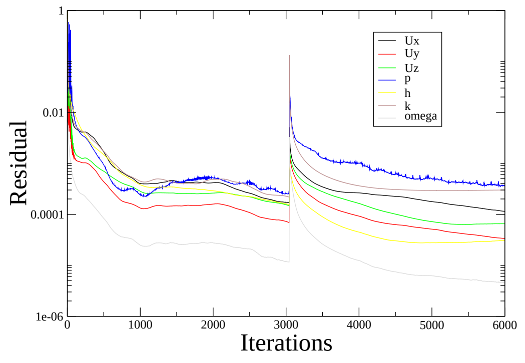
In Figure 6.29 medium valve lifts (4 mm, 5mm, 6 mm) residuals plots are described. Medium valve lifts are characterized by greater stability. Plots show that residuals achieve lower values and present less oscillatory behavior than previous simulations. A common trait to medium valve lifts consists in a more rapid convergence of $U_z$, $U_y$ and $\omega$, where $z$ is the cylinder axis direction and $y$ is the predominant inlet flow direction.

Figure 6.30 represents high valve lifts 7 mm, 8 mm, 9 mm residuals plots. All the features described for medium valve lifts can be applied to this case. In particular, as shown in Figure 6.30b and Figure 6.30c, simulation residuals have an even more linear trend.

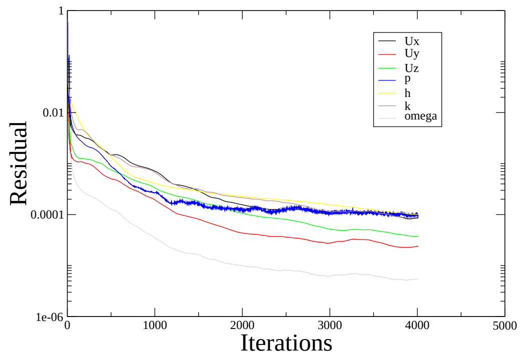
A common aspect to all significant simulations (3mm-9mm) is that pressure residual decreases more slowly than other quantities. This phenomenon can be considered normal using these boundary conditions.



(a) 1 mm valve lift

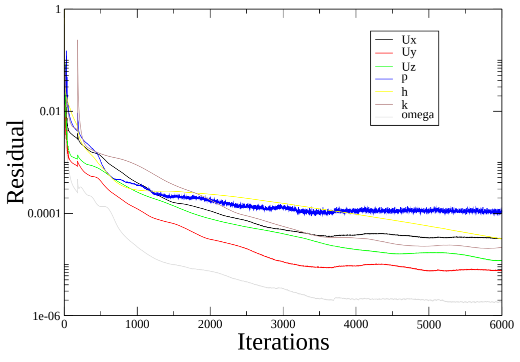


(b) 2 mm valve lift

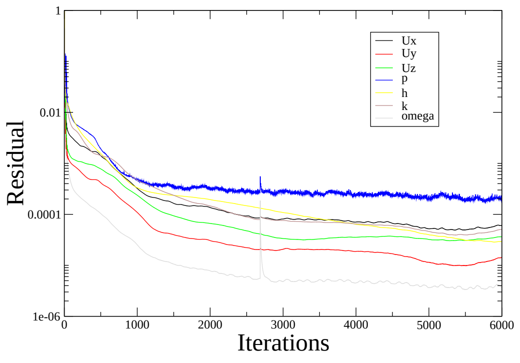


(c) 3 mm valve lift

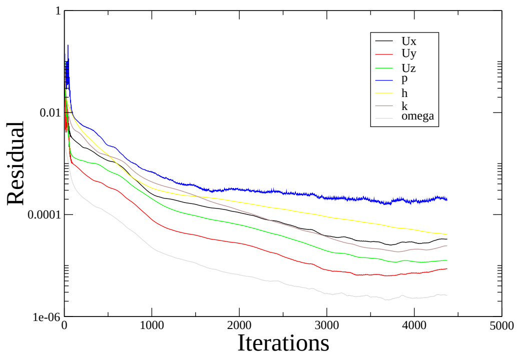
*Figure 6.28: Convergence results for lift 1 - 3 mm*



(a) 4 mm valve lift

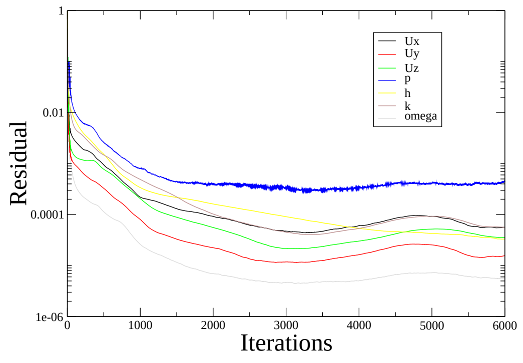


(b) 5 mm valve lift

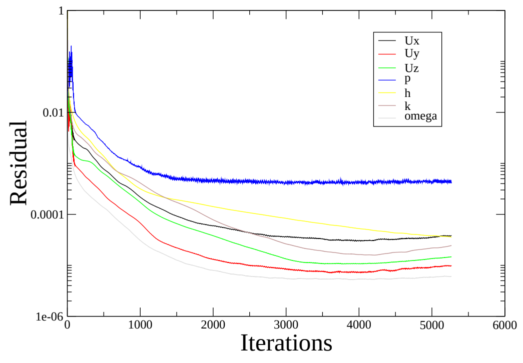


(c) 6 mm valve lift

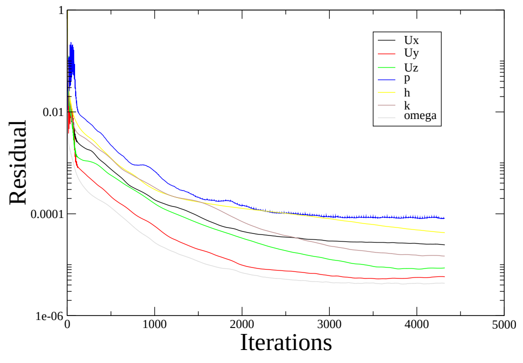
*Figure 6.29: Convergence results for lift 4 - 6 mm*



(a) 7 mm valve lift



(b) 8 mm valve lift



(c) 9 mm valve lift

*Figure 6.30: Convergence results for lift 7 - 9 mm*

# Chapter 7

# Conclusions

The aim of this thesis work is to describe air movement characteristics in cylinder and intake port of a spark ignition engine, analyzing the generation and the evolution of the flow field during the intake stroke at different valve lifts and check the ability to realize arranged motions in the cylinder. Moreover at a later stage it aims at comparing these results with experimental investigations ones.

Thanks to this work, as shown in Chapter 6, it has been possible to describe the air flow behavior within the cylinder during the intake stroke at different valve lifts. Simulation results show that tumble is the main in-cylinder flow motion, and tumble vortex grows in size and intensity by the increasing air entering through the valves. The comparison between numerical and experimental tests have produced comparable results. For intermediate valve lifts (3 mm to 7 mm) the velocity maps are satisfactorily matched. To confirm this also the trend of discharge coefficient and tumble coefficient can be discreetly compared in this range of lift.

The differences in results that it is possible to detect may be due to several factors. It is needed to accept that computational fluid dynamic simulations, in any case, are approximated and limited by the resources availability. Because of poor hardware resources and complex geometry features, it has been necessary to find a compromise between refinement and computational time. With $k$ - $\epsilon$ or its derived turbulence models, coarse grid can be used; however to ensure $y^+ > 30$ the cell size is not compatible with the refinement required in the valve throat zone. Moreover, these turbulence models perform poorly for complex flow involving severe pressure gradient, separation and strong streamline curvature. $k$ - $\omega$ $SST$ turbulence model has proven to be the most suited for this work. Referring to mesh generation, the use of $snappyHexMesh$ as meshing tool has led to issues related to the fact that this utility is based on iterative and automatic meshing processes. The main problem lies in the fact that the angular deviation of the flow directions from the vector connecting the two cell centers could generate numerical diffusion. Designing a mesh with cells oriented in the same flow direction, in valve throat zone, could represent a solution to face this problem. However, adding boundary layer, without finer wall refinement, generates distorted cells with too high non-orthogonality and skewness values. Proceeding with further wall refinement, instead, would involve a doubling of the number of cells that is incompatible with computational resources. A commercial software allows to design a mesh with cells oriented in the same flow direction, without any significant increase in the number of cells. It could represent a possible future development for this work.

Experimental tests cannot be completely reliable for several hypothetical reasons. Indeed, the cylinder outgoing flow tends to expand, losing the ordered structure set by flow straightener and it could be slowed by the resistance that it meets in quiet air outside the cylinder. When the flow velocity is very low, at low valve lift (1 mm and 2 mm), the hot wire anemometer probe is affected by the effect of natural convection and the results might be not reliable. The size of probe and movement system inserted in the flow field may present a certain intrusiveness and affect the results.

In addition, the difference observed in the comparison of the results may be due to the following aspects: an error could occur in dial gauge indicator for valve lift measurement, then there could be discrepancy in the intake pressure measurement location between CFD simulation and flow bench and finally, CAD geometry has been slightly modified compared to flow box geometry tested on the flow bench, in order to adapt it to simulation conditions.

For all these reasons, it is possible to claim that the use of numerical simulations cannot leave aside the experimental tests and vice versa, in order to obtain a complete multi-dimensional steady-state in-cylinder flow study.

# Bibliography

[1] J. H. Ferziger, M. Peric, *Computational Methods for Fluid Dynamics* Springer, 2002.

[2] H. K. Veersteg, W.Malalasekera, *An introduction to Computational Fluid Dynamics,* Technical, 1995.

[3] A. Quarteroni, F. Saleri, *Introduzione al calcolo scientifico,* Springer, 2002.

[4] *OpenFOAM 2.2.0 Programmer's Guide,* February 2013.

[5] Kundu, Cohen, Dowling, *Fluid Mechanics,* Academic Press, 2011.

[6] Menter, F. R., *Zonal Two Equation k-$\omega$ Turbulence Models for Aerodynamic Flows,* AIAA Paper 93-2906., 1993.

[7] Frank M. White, *Text book on Fluid Mechanics,* Fourth Edition, McGraw-Hill Book Company

[8] Ferrari, G., *Motori a Combustione interna,* Il Capitello.

[9] W.P.Jones and B. E. Launder, *The prediction of laminarization with a two-equation model of turbulence,* International Journal of Heat and Mass Tranfer 15.

[10] *OpenFOAM 2.2.0 User Guide,* February 2013.