POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

CFD simulation methodology for Wankel engines

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Author: **Matteo Zanotto**

Student ID: 830039
Advisor: Prof. Tommaso Lucchini
Academic Year: 2022-23

Abstract

This thesis focuses on the development of a methodology suitable for the fluid dynamics simulation of Wankel engines using OpenFOAM 8 software. After an initial introduction to the Wankel rotary engine, primarily focused on small-sized engines for unmanned aerial vehicles (UAVs) and range extenders, the study of the fundamental equations for describing its geometry and kinematics was carried out, followed by the implementation of a procedure for generating the calculation grid. This method was developed for both a simplified single-chamber configuration and a complete three-chamber configuration. The method was implemented by creating new utilities specific to this type of engine, which were subsequently integrated into the OpenFOAM Lib-ICE, a library developed by the ICEgroup research group at Politecnico di Milano. Once the grid generation model was validated, it was used for a preliminary analysis of the engine. However, due to specific issues related to the stability of simulations for this engine, considerable efforts were made to ensure the convergence and stability of these. Therefore, the final part of the work is focused on a gas dynamic analysis of the engine and the differences between the single-chamber and three-chamber configurations, while combustion and injection studies were not addressed. More comprehensive and detailed studies, including thermal modeling of the engine and comparison with additional experimental data, are necessary for a better characterization of the engine.

Keywords: CFD, OpenFOAM, Wankel engine, mesh generation

Abstract in lingua italiana

Il presente elaborato è incentrato sullo sviluppo di una metodologia adatta alla simulazione fluidodinamica dei motori Wankel utilizzando il software OpenFOAM 8. Dopo un'iniziale introduzione sul motore rotativo di tipo Wankel, incentrata principalmente su motori di piccole dimensioni per veicoli aerei senza pilota (UAV) e range extenders, si è passati allo lo studio delle equazioni fondamentali per la descrizione della sua geometria e della sua cinematica, per poi implementare un procedimento per la generazione della griglia di calcolo. Quest'ultimo è stato sviluppato sia per una configurazione semplificata a camera singola, che per una completa, ossia a 3 camere. Questo metodo è stato implementato creando nuove utilities specifiche per questo tipo di motore che, successivamente sono state inserite nella libreria di OpenFOAM Lib-ICE, sviluppata dal gruppo di ricerca ICEgroup del Politecnico di Milano. Una volta validato il modello per la generazione della griglia, quest'ultimo è stato utilizzato per un analisi preliminare del motore. Tuttavia, a causa di problematiche specifiche di questo motore relative alla stabilità delle simulazioni, numerosi sforzi sono stati impiegati per assicurare la convergenza e la stabilità di queste ultime. Per questo motivo, la parte finale del lavoro è focalizzata su un'analisi gasdinamica del motore e sulle differenze tra la configurazione a singola camera e quella completa, mentre la combustione e lo studio dell'iniezione non sono stati trattati. Studi più completi e dettagliati, che includano anche la modellazione termica del motore e il confronto con ulteriori dati sperimentali, sono necessari per una caratterizzazione migliore di quest'ultimo.

Parole chiave: CFD, OpenFOAM, Wankel engine, Mesh generation

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

Introduction **1**

1 Wankel engine **3**

1.1 General Overview of Rotary Engines . . . 3

1.1.1 Structure . . . 4

1.1.2 Working Principle . . . 5

1.1.3 Components . . . 7

1.1.4 Advantages . . . 10

1.1.5 Disadvantages . . . 12

1.2 Historical Background . . . 15

1.3 Applications of Wankel Engines . . . 19

2 CFD Basics **23**

2.1 OpenFOAM introduction . . . 23

2.1.1 OpenFOAM environment . . . 23

2.1.2 Lib-ICE . . . 25

2.1.3 Case organization . . . 27

2.2 Fluid-dynamic equations . . . 28

2.2.1 Continuity equation . . . 29

2.2.2 Momentum equation . . . 30

2.2.3 Energy equation . . . 30

2.2.4 Constitutive laws . . . 31

2.3 Finite Volume method . . . 32

2.3.1 Solution domain discretization . . . 32
2.3.2 Equations discretization . . . 34
2.4 Linear equation system . . . 42
2.4.1 Matrix assembly . . . 42
2.4.2 Boundary conditions . . . 44
2.5 Turbulence modeling . . . 45
2.5.1 RANS . . . 47
2.6 Solution strategies . . . 50
2.7 Grid motion . . . 52

3 Grid generation with OpenFOAM 55
3.1 Equations . . . 55
3.1.1 Stator shape . . . 57
3.1.2 Rotor shape . . . 60
3.1.3 Oscillation angle . . . 62
3.1.4 Rotor motion . . . 62
3.1.5 Engine Displacement Law . . . 63
3.1.6 Compression Ratio . . . 63
3.2 Utilities . . . 65
3.2.1 `createWankelEngineMesh` . . . 65
3.2.2 `makeWankelMesh` . . . 66
3.2.3 `makeWankelFullMesh` and `createWankelEngineFullMesh` . . . 69
3.2.4 `createWankelIntakeExhaust` . . . 70
3.3 Dictionaries . . . 71
3.3.1 *wankelGeometry* . . . 71
3.3.2 *meshParameters* . . . 71
3.4 Additional utilities and dictionaries . . . 74
3.4.1 `blockMesh` and `blockMeshDict` . . . 74
3.4.2 Utilities and dictionaries related to ACMI coupling . . . 77
3.5 Model consistency . . . 80
3.5.1 AIE 225CS . . . 80
3.5.2 Engine displacement law and CR . . . 81

4 Engine analysis 85
4.1 Case setup . . . 85
4.1.1 Mesh setup . . . 87
4.1.2 Boundary and initial conditions . . . 89
4.1.3 Turbulence model . . . 95

4.1.4 Solver and numerical setup 98
4.2 Mesh sensitivity analysis 104
4.3 Gas Dynamic comparison 111

5 Conclusions **127**

Bibliography **129**

List of Figures **133**

List of Tables **137**

List of Acronyms **139**

List of Symbols **141**

Acknowledgements **143**

Introduction

The Wankel rotary engine, first developed by Felix Wankel in the 1950s, represents an innovative approach to internal combustion engines that offers several advantages over traditional reciprocating piston engines. Despite the disappearance of rotary engines as standalone power sources in automotive applications due to emission regulations, the technology continues to be explored for other purposes. In fact, Wankel engines are known for their compact size, lighter weight, and smooth operation, which make them particularly suitable for specific applications such as small scaled Unmanned Aerial Vehicles (UAVs) and range extenders in hybrid electric vehicles. As the electrification of transport continues to progress, hybrid electric vehicle (HEV) and range-extended electric vehicles (REEV), employing small-sized rotary engines, can help bridge the gap between conventional internal combustion engines (ICE) and fully electric vehicles by offering some advantages of electric propulsion without the issue of range anxiety [1] [2]. Moreover, Wankel engines, with their unique design and operational characteristics, have emerged as a promising solution also for UAVs and aerospace applications, where size, weight, and vibration are critical factors. Wankel engine demonstrated to be a good candidate also thanks to its multi-fuel capability, since, in this field of application, propulsion systems capable of running on heavy fuels, like jet fuel or kerosene, are often desired or mandatory due to fire safety regulations and availability [3].

Despite the potential benefits, Wankel engines also present some challenges, such as lower fuel efficiency and higher emissions compared to conventional engines. To fully exploit the advantages of Wankel engines while addressing their limitations, it is crucial to develop and refine computational models that can accurately simulate the complex processes occurring within these engines.

In light of this context, the primary objective of this thesis is to develop a computational fluid dynamics (CFD) methodology for simulating Wankel engines with the goal of investigating the fluid dynamic fields and performance characteristics. To achieve this, the open-source software OpenFOAM 8 was utilized, and a mesh generation procedure was implemented using its capabilities and versatility. This was accomplished by developing new utilities through custom C++ code, which were subsequently integrated into the

Lib-ICE, an OpenFOAM library created by the ICEGroup at Politecnico di Milano.

A comprehensive study was conducted on various equations describing the geometries and kinematics of rotary engines, enabling the development of the mesh configurations for the simulations. Two different mesh configurations were generated for this study: one simulating a single rotor flank (one single chamber), and another simulating three rotor flanks (three chambers).

However, due to the challenges faced in achieving convergence for the simulations, a significant portion of the work focused on modifying the numerical setup and addressing specific problems unique to this type of engine. As a result, combustion and spray modeling were not within the scope of this study. Instead, a preliminary gas dynamic analysis was performed using iso-octane as a fuel and comparing the results between the two configurations (one and three rotor flanks).

For the 1-chamber configuration, a mesh sensitivity analysis was conducted by varying the duct lengths for both intake and exhaust, and evaluating the effects on chamber filling.

In order to effectively analyze and visualize the results obtained from the simulations, the postprocessing and analysis were conducted using the ParaView software, integrated within the OpenFOAM framework, as well as MATLAB.

Through this work, valuable insights into the challenges and complexities of simulating small Wankel engines have been gained. Future research can build upon these findings to further refine the CFD methodology and eventually incorporate combustion and spray modeling to create a more comprehensive simulation framework for this type of engine.

The thesis work is structured as follows: Chapter 1 provides an overview of the Wankel engine's history, design, and applications, together with the advantages and disadvantages of its layout. Chapter 2 introduces the OpenFOAM environment, presents the fundamental equations of CFD, their discretization, and explains the methodology used by the finite volume method, from the equations to the solution algorithms. Chapter 3 delves into the geometries and kinematics of the Wankel engine, introduces the mesh generation utilities, explains their working principles, and evaluates the consistency of the volume law, quantifying the interpolation error and the discrepancy generated by the model assumptions. Chapter 4 presents the results of the simulation, including the mesh sensitivity analysis, and a preliminary gas dynamic comparison between the 1-chamber and 3-chamber configurations. Finally, Chapter 5 summarizes the main findings, discusses the implications for Wankel engine development, and suggests avenues for future research.

1 | Wankel engine

In this chapter the main features of a rotary engine are presented, as well as some historical background.

1.1. General Overview of Rotary Engines

Wankel engine, despite its fame in the automotive industry, does not have a unified definition. According to Mazda's engineer Kenichi Yamamoto, in the book "Rotary Engine", it is defined as: *"An internal combustion engine that performs the four strokes of intake, compression, expansion and exhaust while the working chamber changes its volume and the moving parts always rotate in the same direction"* [4]. From this definition it is straightforward to understand that these machines differs from other conventional engines mainly for the absence of reciprocating parts, allowing them to have a relatively compact and simple design, with small number of components (see Figure 1.1). The following paragraphs will present a more detailed overview of the topic, including the main implications of its peculiar design.

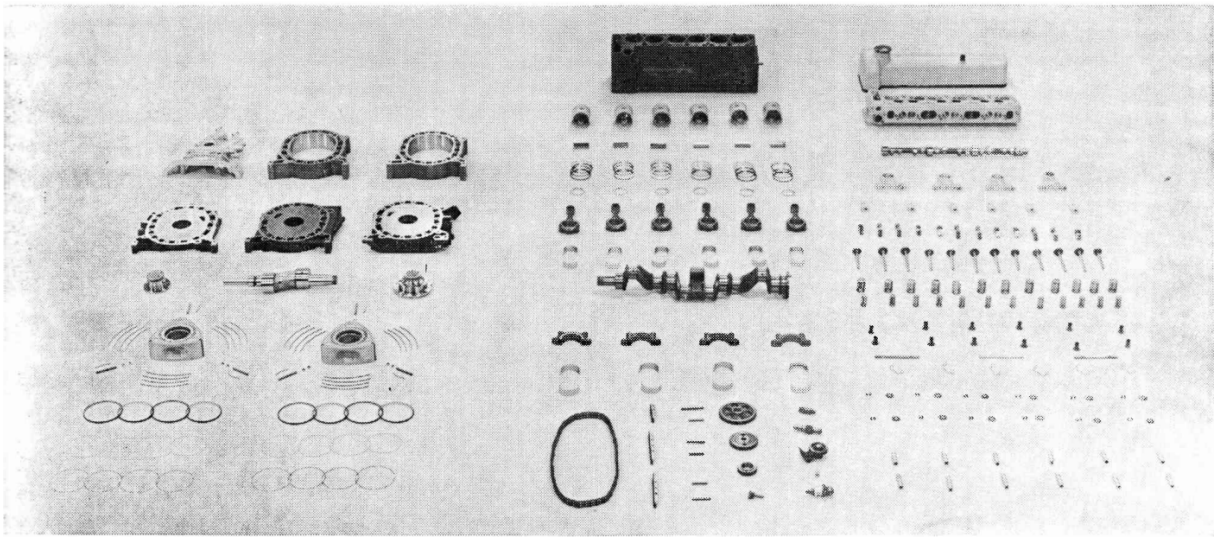


Figure 1.1: On the left: Wankel engine components. On the right: Inline-6 Piston engine components [4].

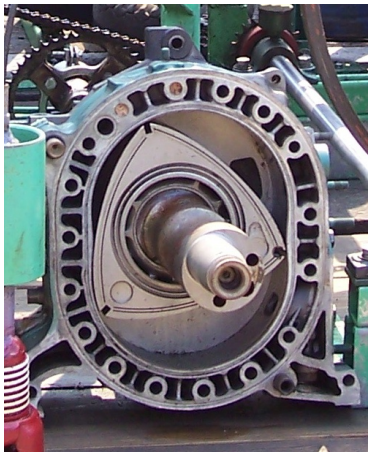
1.1.1. Structure

The basic morphology of the engine is shown in Figure 1.2. As it can be seen, the internal shape of the housing has a peculiar shape, derived from a geometric figure called peritrochoid (see 3.1), which corresponds to the cylinder in a usual piston engine, and a triangular-like shape rotor, that corresponds to the piston. On the rotor lateral surfaces some recesses are present, which shapes can change in order to obtain different type of combustion, spray impingement and regulate the CR. The rotor identifies three chambers that are working together simultaneously, carrying out different phases of the complete cycle. The phasing gear is composed by one stationary side, fixed on the side housing, and one fitted on the rotor's hole for a finite fraction of its internal surface. This gear system has a gear ratio of 3:1 and it is responsible for the rotating motion of the rotor around its axis, while the eccentricity of the output shaft creates the rotation of the rotor axis around the shaft's one. This motion allow the tips of the rotor (where the apex seals are located) to draw the internal shape of the rotor housing, keeping them always in contact with the stator and making possible the sealing between the chambers. The rotor housing features a cooling system, an exhaust port and the holes for the spark plugs[1], which can be glow plugs or absent in case of compression ignition (CI) REs, and in variable number. Usually it can contain also the intake port (peripheral porting), which in case of Figure 1.2 is located on the side housing (side porting). The location of the intake port has some performance and design implications, because it can regulate the overlap time. Additional observation that are worth mentioning [5]:

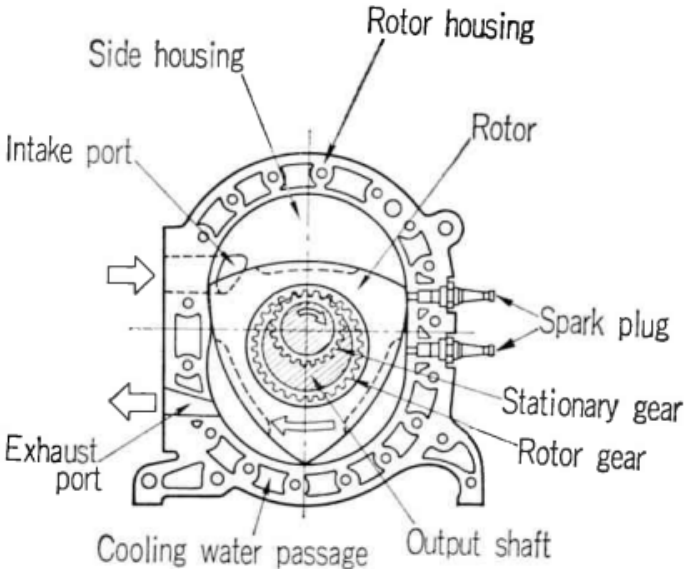
- the unique kinematics of the rotor resulting from the peritrochoidal configuration affects the length of the cycle, that in Wankel engine is 1080° instead of the classical 720° or 360° for the four and two-stroke engines respectively;
- there is a 3:1 ratio between the eccentric shaft and the rotor angular displacement and a linear relation exists between them.
- the rotor can be inscribed in a circle with the apices lying on the circumference. The angular distance between two apices is 120°. As a consequence of the previous consideration, the rotor will assume the same configuration inside the housing with a period of 360° of the eccentric shaft;
- a point placed on the peritrochoid will be exposed to each flank of the rotor for 360° of the eccentric shaft angle as a consequence of the angular displacement of the apices;

[1]The tip of the spark plug in Wankel engine cannot enter the chamber and it resides in a small recess of the rotor housing

- a peripheral port, i.e. the inlet/outlet ports of the engine will be exposed to each flank of the rotor for a total of 360° of the eccentric shaft angle added to the angular extent of the port given by the angular distance of the two extreme points on the peritrochoid.



(a) Wankel-section of Mazda 13B engine [6]



(b) Wankel structure from [4]

Figure 1.2: Section and main parts of the engine.

1.1.2. Working Principle

The Wankel engine is a four stroke engine[2] that is usually performing an Otto Cycle, but recently, thanks to some designs that remove port overlap and the use of turbocharging, it can perform Atkinson and Miller cycles. Figure 1.3 shows the main phases of the engine. Starting from 1, we have the intake port opening (IPO) and the intake phase begins in the relative chamber, with the introduction of the fresh mixture. In this specific case there is no or little overlap between the opening of the 2 ports, but it often happens, especially in peripherally ported engines to have some non negligible effects of the ports interaction, since the chamber is facing both of them simultaneously (see 4.3). The chamber will expand (2, 3) till its maximum volume (4), before starting the compression stroke. In 5 the intake closes (intake port closing (IPC)) and the chamber is completely sealed, starting moving towards reduced volumes 6 and 7. In 9 we have the smallest volume, which corresponds to top dead center (TDC), and the ignition of the air-fuel mixture by the

[2]Even if it has some peculiarities that makes it a sort of hybrid between 4-stroke and 2-stroke engines.

spark plugs. The subsequent expansion stroke is shown in points 10,11 and 12, while the exhaust port opening (EPO) starts the exhaust phase in 13. The exhaust phase is shown in points 14, 15, 16, 17 and finally 18, with the consequent exhaust port closing (EPC). A comparison with the conventional engine strokes are shown in Figure 1.4. Some other interesting observations can be made:

- since for one rotor revolution the output shaft undergoes three revolutions and all the chambers undergoes their 4-stroke cycle, three ignition events are happening during this period (1080 crank angle dregree (CAD), or eccentric shaft angle (ESA)). This means that, similarly to 2-stroke engines, the Wankel engine undergoes one combustion event for every output shaft rotation;
- the time needed to complete one of the strokes is 270 ESA, 1.5 times longer than conventional piston engines (180 CAD). This due to the fact that to complete the four strokes of one chamber 3 output shafts rotations (1080 ESA) are needed.

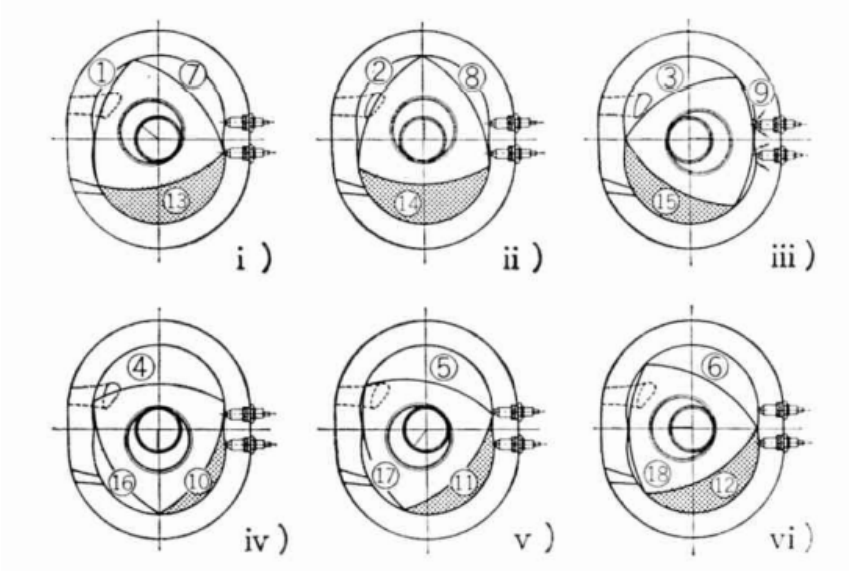


Figure 1.3: Engine strokes [4].

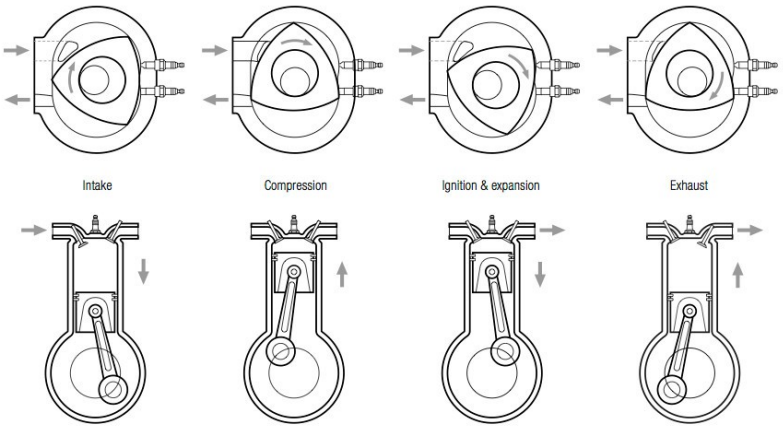


Figure 1.4: Comparison with conventional engine strokes [7].

1.1.3. Components

Here below a more exhaustive presentation of the main components is presented:

- **Rotor** The rotor is the equivalent of the piston in conventional engines and is responsible to transmit the power trough the output shaft. The rotor also plays the role of intake and exhaust valve, because with its motion covers and discloses the ports. It has three sides, named flanks, with a rotor recess on it. The rotor recess is an important element in the design because it influences heavily the motion and the combustion inside the chamber. The rotor land, instead, is a small region used for lubrication in the zone in contact with the side housing. The rotor in a Wankel engine is typically made of a lightweight and strong material, such as aluminum or an aluminum alloy. This is because the rotor rotates at very high speeds, up to 7000 RPM or more, and therefore it must be able to withstand the high centrifugal forces and stresses generated during operation. Additionally, the low weight of the rotor is advantageous for reducing the overall weight and improving the performance of the engine. In some cases, the rotor may be coated or treated with a ceramic or other high-strength material for increased durability and wear resistance. The rotor has the gear incorporated and the sealings, specifically the apex seals, the side sealings and the oil seal.

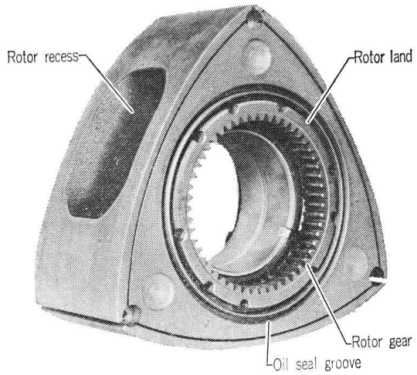


Figure 1.5: Rotor picture from Mazda[4]

- **Rotor housing** The rotor housing is the equivalent of the cylinder and its internal shape fits the motion of the apex seals. It contains the channels for the liquid or air cooling, the recesses for the spark plugs and the intake and exhaust ports. The material must have high strength to withstand the pressure peaks, small thermal expansion to maintain the perfect clearances, good thermal conductivity and wear resistance to endure the friction of the apex seals. For this reasons, cast iron is good for thermal expansions and cost, while gamma silmin aluminum alloy is good for the weight. In this case, the internal surface of rotor housing is made by hard chrome plating containing silicon carbide particles or molybdenum alloys, which is then placed on a steel sheet created with sheet metal insert process (SIP). This process creates a fine saw-tooth surface on one side to bond better with aluminum of the housing. One more advanced alternative is to spray coating materials like CerMet (composed by a compound of ceramic plus metallic materials) on the rotor housing/side housing surfaces and creating porous surfaces to improve oil lubrication.

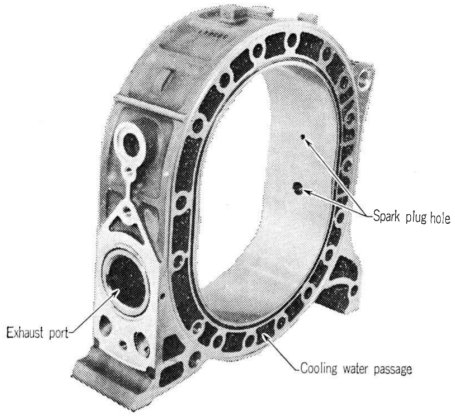
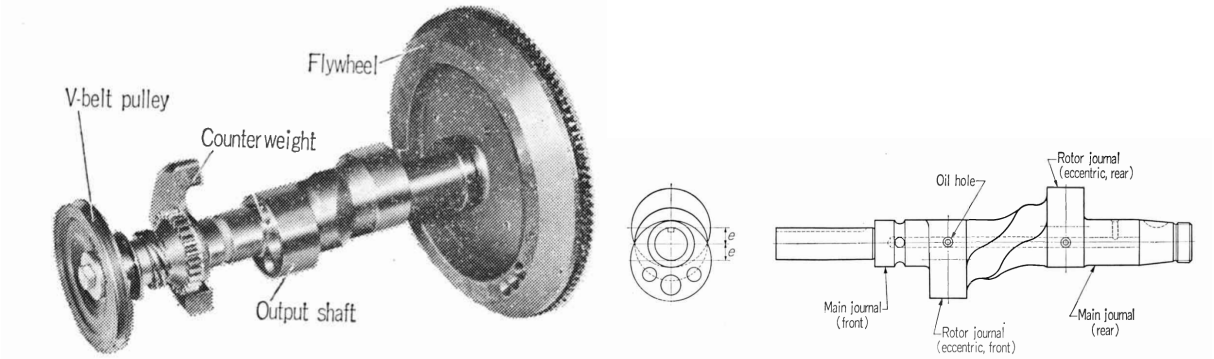


Figure 1.6: Rotor housing picture from Mazda [4]

- **Eccentric shaft** The eccentric shaft plays the same role of the crankshaft in conventional piston engines. As shown in Figure 1.7 the axis of the journal is eccentric of a distance e with respect to the shaft axis to ensure the rotation of this one trough the action of the rotor motion. It incorporates oil holes for lubrication, weights for balance and, as usually, a flywheel. It is possible to balance completely the shaft inertia forces with a properly tuned counterweight system, since the only contribution to eliminate is the one given by the rotating mass. The shaft is supported by two bearings fixed on the stationary gears, placed on the side housing. The shaft is usually made of forged chrome steel or chromemolybdenum steel, to avoid bending and torsion problems.



(a) (b)

Figure 1.7: Shaft picture from [4]

- **Sealings mechanism** The sealings are crucial part of the Wankel engine, and it has been a bottleneck and a subject of many studies of its design for long time. The sealings are shown in Figure 1.8 and as it can be seen they are divided in apex seals, sides seals and corner seals. The first one are located on the top of the rotor tips and are responsible of sealing one chamber respect to the adjacent ones and to keep the gas-tight. This means that the apex seals will face pressure and thermal gradients cyclically, causing it to oscillate laterally inside its seat towards the chamber with lower pressure. To ensure the sealing, a spring supports each apex seals. Instead, to avoid wear problem, that can be caused by manufacturing errors or thermal expansion, a clearance is required and a split-type design is used. This also helps to avoid the formation of chatter marks on the rotor surface, because this design changes the resonating frequency of the seals, which was causing this phenomenon. For all these reasons also the material choice is critical; nowadays, surface treated advanced materials are used, e.g. cast iron as base metal chilled by electron beam, acting over the already mentioned porous rotor housing, or special synthered alloys

on a Ni-SiC plated rotor housing. The side seals instead avoid the leakage of gases inside the side of the rotor, and, since in its motion it passes trough the locus of the oil seal, it has a better lubrication with respect to the apex ones. Usually a sintered alloy or a special cast iron is used. These material are used also for the corner seals, with the addition of a chrome plated surface to resist wear, which function is to keep the gas-tight between the apex and the side seals.

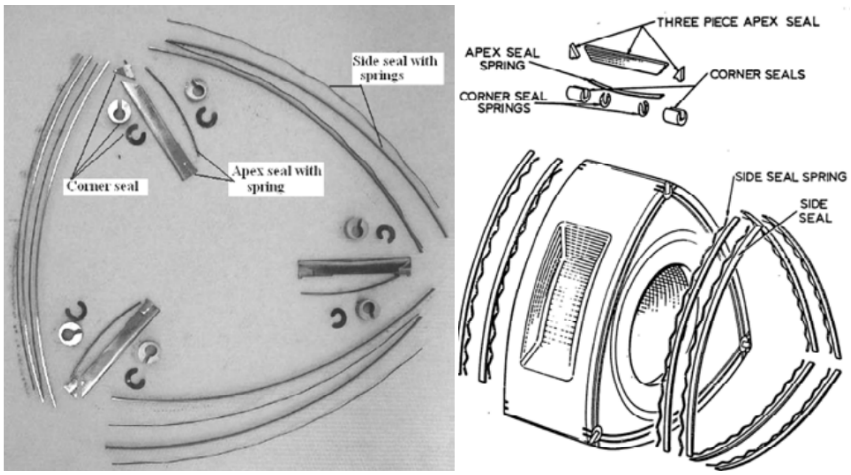


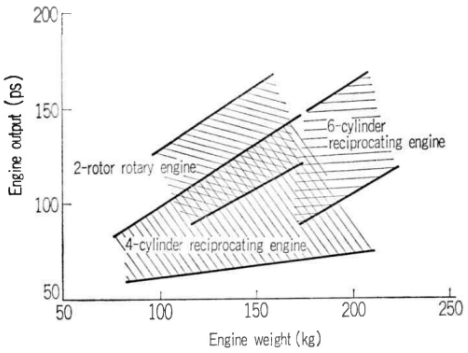
Figure 1.8: Sealings of the rotor from [8].

1.1.4. Advantages

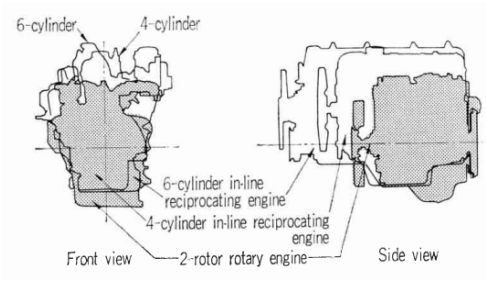
In this paragraph the main advantages of the rotary engine are presented. Most of them are related to the absence of reciprocating parts and the considerations made previously on its structure.

- **Smoothness** The engine is characterized by continuous unidirectional motion and absence of the inertia of reciprocating parts, allowing it to reach high revolutions per minute (rpm). It is easy to be balanced and the power pulse are one per shaft rotation (contrarily to 4-stroke piston engines).
- **High volumetric efficiency** True also at higher loads due to the absence of valve restriction and the longer intake stroke (270 ESA). This peculiarity improves mixing time and allows for charge stratification, in fact the rotor rotates slower than the output shaft, so the effective rpm are lower.
- **Simple design** There is no valve mechanism, so the correct timing for opening and closing can be maintained even at high rpm, and the number of parts is reduced with respect to piston engines.

- **Multifuel capability** The engine has a separate combustion region from intake region, preventing the formation of local hotspots, that could cause preignition or detonation, allowing more flexibility in fuel choice.
- **High Power density** This means high Power/Weight and Power/Volume ratios. Figure 1.9a shows how Wankel engines performs with respect to 4 and 6-cylinder engines.
- **Compactness** Figure 1.9b shows a size comparison with an inline 6 engine. The engine occupies $\frac{1}{3}$ of output speed shaft.



(a) From [4].



(b) From [4].

Figure 1.9: Size and specific power comparison.

- **Lightweight** Due to its dimensions and simplicity.
- **Low vibrations** Extremely low torque oscillation, since there are three power strokes per shaft revolution.
- **Low noise** Due to the lower burning rate and reduced mechanical noise.
- **Perfect balance** The Wankel engine has a better state of balance than a reciprocating piston engine due to the nature of its design. Its rotor is a perfectly balanced point mass that rotates uniformly on an eccentric shaft, making the engine configuration balance similar to the one of a turbine. This design allows for simple bobweights to be used to balance the engine, resulting in an excellent state of balance for both primary and second forces. While some reciprocating engines are considered to have near-perfect balance, such as the inline 6-cylinder engine operating on the 4-stroke cycle, they are not as balanced as a purely rotary engine [9].
- **Low mechanical loss** The Wankel engine's sealing grid exhibits remarkably low friction, which may seem paradoxical when compared to the piston ring pack of a

reciprocating engine. However, this is because the rotor and sealing grid rotate at a third of the engine speed and do not undergo directional changes like in a reciprocating engine. Furthermore, a single rotor's sealing grid replaces the function of three cylinders in a reciprocating engine, thereby inherently reducing friction [9].

- **Flat torque curve** thanks to the longer expansion stroke. This allows linear power delivery for improved drivability[3].
- **Simple manufacturing** Lower production costs and easier scalability, because it is sufficient to increase the number of rotors to increase the power output.
- **Prone to Turbocharging** Thanks to the low compression ratio (CR) and the high level of exhaust energy, especially for peripherally ported engines.

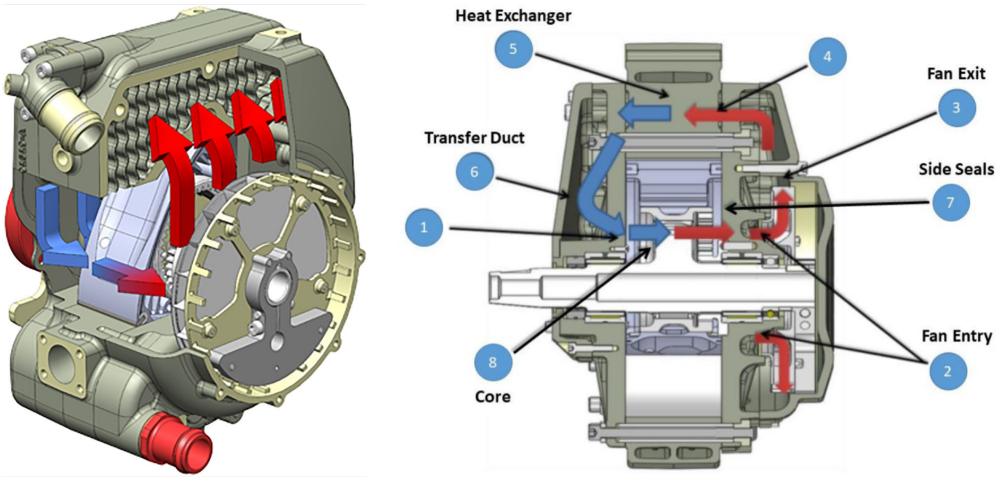
1.1.5. Disadvantages

In this paragraph the main disadvantages are listed, however is important to point out that with new technologies and improvements many of them can be overcome.

- **High fuel consumption** True especially at low and high loads due to the slow combustion speed.
- **High oil consumption** Due to the total loss lubrication system, since oil must enter directly in the chambers to lubricate the contact surface between apex seals and rotor housing, and it burns causing aditional hydrocarbon (HC) emissions. However, it is worth mentioning that latest unmanned aerial vehicle (UAV) engines employ a lubrication system that can limit oil consumption to $10 cc/hour$[10].
- **High emissions** Despite in recent years some ways to mitigate this problem have emerged, Wankel engine produces more pollutants, especially HC, due to the total loss lubrication system, that is often used. The NOx emissions, instead, can be lower due to the lower flame temperatures reached in the chamber.
- **Blow-By due to sealings** Reaching the perfect seal between chambers is an hard task in Wankel engines because many sealed interfaces are present. New ways of using this blow-by gases are being developed, e.g. the self-pressurizing air-cooled rotor system (SPARCS) system [5], which is a system that uses combustion gas blow-by from the combustion chamber to pressurize the core of the engine (inside the rotor hole, where the shafts is located) to pressurise the internal core of the rotor. The gas blow-by pass trough the properly design side seals due to the pressure difference

[3]This is not always true in peripherally ported engines.

induced by the combustion, until a pressure balance is reached. Then, thanks to a centrifugal fan they are moved trough a closed loop circuit into an heat exchanger (included in the stator), before entering again in the rotor to cool it down. This has a positive effect since in a Wankel rotary engine, the blow-by gas and subsequent core pressure acts equally around the rotor, with little impact to engine brake mean effective pressure (BMEP), contrarily to a traditional reciprocating piston engine, where blow-by gas causes pressure within the crankcase to rise and impact engine performance through increased crankcase pumping work. Other producers like Mazda, instead, in the 13B-MSP RENESIS, developed a new cut-off seal to eliminate the the blow-by of gases between the intake and exhaust ports via the slight gap between oil seals and side seals. This is able to impede gas carryover in the subsequent cycle.



(a) (b)

Figure 1.10: SPARCS System schematization from [11].

- **Low CR** Typically it is very limited (below 13:1) in a rotary engine (RE) mainly because of the geometric shape of the trochoid that has a small displacement volume. Another cause of the reduction of the effective CR is the imperfect sealing of the apex seals.
- **Drivability at low loads** Affected in peripherally ported engines due to high residual gas carryover.
- **Uneven distribution of thermal load** The combustion for all chambers happens in the same angular position, leading to an asymmetry in temperature distribution and deformation along the rotor housing.

- **Poor combustion chamber shape** The geometry of the chamber is not optimal for combustion and, despite the strong squish given by the rotation, is not able to create a strong mixing of the air-fuel mixture, causing the combustion process to be slower and reducing thermal efficiency. The chamber has a high surface to volume ratio, leading to heat losses, that are the main cause[4] for flame quenching [12]. Also, the chamber is long and narrow and the travel time of the flame front is long. Figure 1.11 shows a possible model of flame propagation for two-spark-plug rotary engine, highlighting four flames fronts and the unburned regions. Figure 1.12, taken from [13], instead, shows the firing pressure curve of the advanced innovative engineering (AIE) 225CS engine for different BMEP values. This picture is hereby reported to highlight the peculiar trend of the pressure in a Wankel RE, that, due to the slow combustion, rises slowly after TDC.

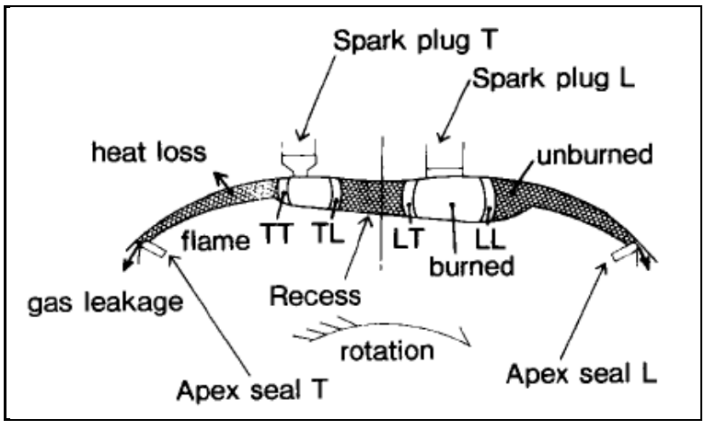


Figure 1.11: Flame propagation model for two spark plugs rotary engine [14].

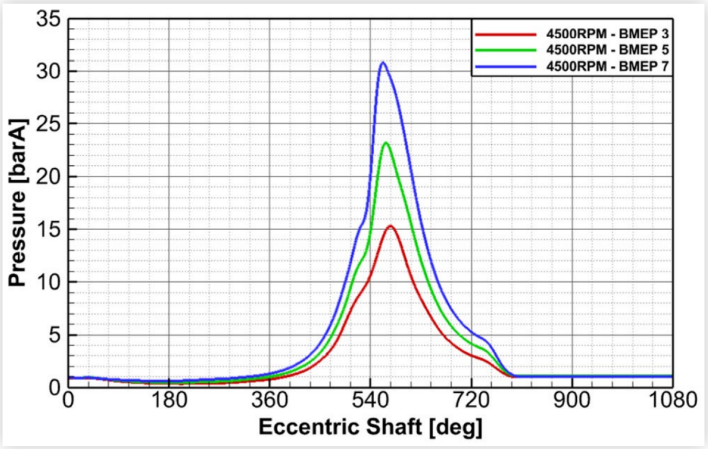


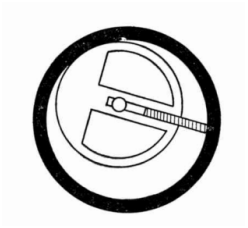
Figure 1.12: Pressure of AIE 225CS for different BMEPs [13].

[4] more than the air fuel mixture quality.

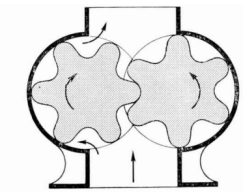
1.2. Historical Background

As one can imagine, the "father" of the Wankel engine is the German engineer from whom it takes its name, Felix Wankel (13 August 1902 - 9 October 1988), who prototyped the first one in 1954, working for the NSU (Neckarsulm Strickmaschinen Union, currently Audi) Motorenwerke AG. However, the foundations for its conception were laid during the preceding years, if not centuries, and its subsequent development, up to the present day, has also been contributed by other engineers and automakers. Among the most important is the Japanese company MAZDA. Below is reported a chronological summary of the evolution of this engine, from the XVI century to the present day.

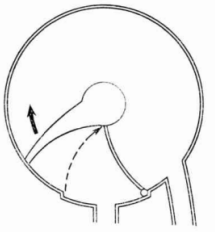
1588-1943 We see the birth of the first concepts of rotary machines, starting with compressors and pumps, then for increasingly sophisticated steam engines and finally internal combustion engines. However, these machines were often unreliable and very complex, despite the advantages of the rotary design. Some example are presented in Figure 1.13.



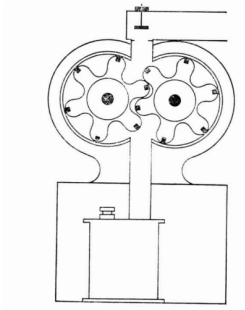
(a) Ramelli's pump.



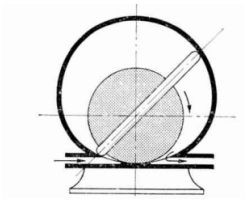
(b) Pappenheim's gear pump.



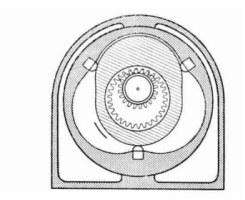
(c) Rotary steam engine by Watt.



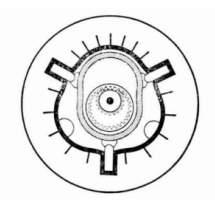
(d) Murdock's rotary steam engine.



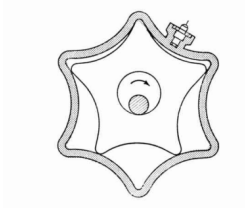
(e) Franchot rotary compressor.



(f) Cooley's rotary steam engine.



(g) Umpleby's Rotary engine.

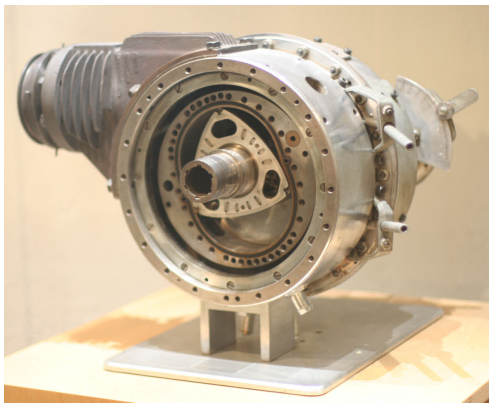


(h) rotary engine by Wallinder and Skoog.

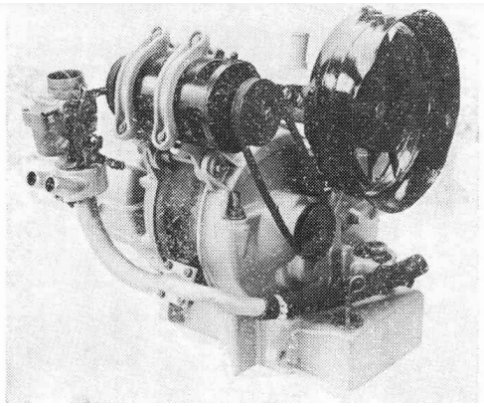
Figure 1.13: Old concepts from [4]

1951 Felix Wankel manufactured a rotary compressor, that was installed as a supercharger on a NSU's motorcycle and established a speed record in the US.

1954-1957 Felix Wankel, starting from the compressor design, created the first Wankel engine prototype, called DKM54 (*DrehKolbenMotor*[5]), capable of 29 CV at 17000 rpm, despite its small displacement (125cc) and diameter (26 cm) (Figure 1.14a). However, this preliminary design wasn't sufficiently reliable and it was abandoned (only 4 pieces were manufactured) in favour of the KKM (*Kreiskolbenmotor*[6]) design, namely the Wankel engine as it is know today. It was created with the help of another engineer named Hanns Dieter Paschke, which performed what he termed a "kinematic inversion", that allowed a new arrangement in which the housing remains stationary and the rotor rotates on the eccentric shaft, with its power output delivered via a conventional flywheel/clutch arrangement. The KKM prototype, with a displacement of 250 cc, develops a power of 30 hp at 5,000 rpm, significantly lower performance than the previous one.



(a) DKM54 engine [15].



(b) KKM engine from [4].

Figure 1.14

1959 After the successful completion of a 100-hour durability test, the NSU-Wankel type rotary engine, featuring a 250cc x 1 rotor, was declared a success. The report of this achievement, titled "The Rotary Engine Achieves Success," was published worldwide, leading many manufacturers to buy licenses in the following years (to cite some of them: Rolls-Royce, Daimler-Benz AG, Alfa Romeo, Porsche, Toyota, John Deere).

1962 NSU unveiled a 150cc x 1 rotor rotary engine, mounted on a watercraft for water-skiing, which incorporated a water-cooling system.

1963 Toyo Kogyo (the contemporary Mazda), after buying the license from NSU, pro-

[5]Literally "Rotary Piston Engine".
[6]Translates to "circuitous piston engine".

duced a prototype car, Cosmo Sports, with a 400cc x 2 rotary engine (L8A), and displayed the motor parts for the first time at the Tokyo Motor show. This engine was the result of the hard work of the head engineer Kenichi Yamamoto (author of [4]) and his team of 47 engineers, that were nicknamed "47 Ronin" for their perseverance and loyalty to the company [16]. Their contribution allowed Mazda to remain independent from bigger manufacturers like Nissan and Toyota, bringing global attention to a still relatively small brand. One of the biggest challenges for the time, considering the little knowledge about composite materials, was the elimination chatter marks (nicknamed "devil's fingernails marks" by enthusiast) generated on the internal surface of the rotor housing, caused by friction and heat. After many trials with exotic materials, the problem was solved by using a hollow shape design of the apex seals, able to change their resonating frequency.

Figure 1.15: Kenichi Yamamoto, his team and the Cosmo Sports Cars.

1964 NSU introduced the sports car Spider, with a 497.5 x 1 cc engine, which was the first rotary engine car on the market.

1967 Also Tokyo Kogyo (Mazda) entered the market with the Cosmo Sports (L10A engine) and NSU introduced a new car, the Ro 80. It is worth mentioning tht in this period some companies, like Curtiss-Wright Corporation and John Deere, Inc., developed the so called stratified charge rotary engine (SCRE) for the general aviation community, that tried to use heavy fuels instead of gasoline.

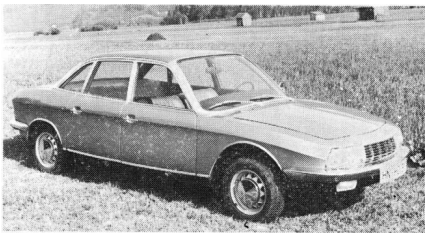


Figure 1.16: Ro 80 (NSU)

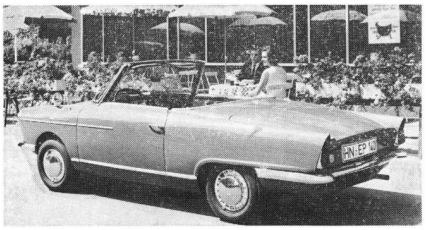
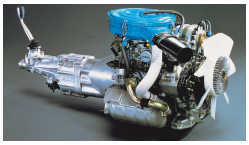
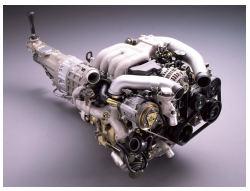


Figure 1.17: Spider (NSU)

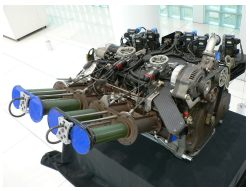
1970-2012 Mazda continued to improve the rotary engine design in the following years and was the only manufacturer to mass produce cars with this kind of engine. One Engine that is worth mentioning is the 12A, due to the introduction of the SIP (Sheet-metal Insert) process, that employs a specially treated steel sheet (similar to a liner found in a standard piston engine). The sheet had a surface coated with chromium to enhance its durability and reliability, making possible to the old carbon seals in favour of conventional cast iron. Another innovation was the 6PI (6 port induction), that allowed for better fuel efficiency and emissions control, as well as improved performance, compared to the earlier 4-port design. It was the first mass produced rotary engine and it was be present in almost every Mazda car of the period. The second iconic engine from Mazda is the 13B (654cc x 2), since it is the most widely produced rotary engine and remained in production for more than 30 years. The most important redesigns of this engine are the 13B-REW and the 13B-MSP RENESIS. The first one features two sequential-turbochargers, giving the highest power to weight ratio. It was mounted on all the Mazda RX-7 models till 2002 and it reaches 280 hp as output. It also made possible for Mazda to win, in 1991 with the 787b car, the 24 hours competition in Le Mans, using a heavily modified version of the engine with 4 rotors (R26B). The 13B-MSP (Multi-Side Ports) RENESIS instead was mounted on the RX-8 starting from 2004 and its goal was to improve fuel economy and reducing emissions adopting some major changes, like side porting (opposing to peripheral), to eliminate overlap, and improved rotor sealing. Advantages and drawbacks of this configuration are extensively covered in [17], [18], [9] and [10]. This allowed the engine to win many rewards in the first years from its debut, however due to more restrictive regulations Mazda discontinued the RX-8 production in 2012.



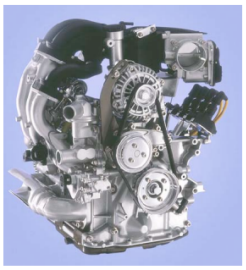
(a) 12A engine.



(b) 13B engine.



(c) R26B [19].



(d) RENESIS engine[17].

Figure 1.18: Most iconic rotary engines from Mazda.

2012-2023 In recent years, Wankel engine has been abandoned in the automotive sector in its use as main propulsion system, but due to its qualities, such as the high power-to-weight ratio, it is finding a renewed market in the production of small aircrafts, UAVs, as well as in its use as a range extender for range-extended electric vehicles (REEV). This is demonstrated by the launch of the new 2023 Mazda MX-30, which features a small Wankel with the goal of recharging a lighter battery.

1.3. Applications of Wankel Engines

As anticipated in the previous paragraph, nowadays, the main applications for Wankel engine are range extenders for electric vehicles and UAVs. Starting with the first one, a brief introduction on the current automotive market should be presented. The push for the decarbonization aim at obtaining a full fleet of full battery elctric vehicles (BEV) and fuel cell electric vehicles (FCEV) by 2050 [20], however the same roadmaps recognizes that internal combustion engines, in the form of hybrid electric vehicles (HEV) and REEV, will have a fundamental role in supporting this progression. As a matter of fact the complete electrification could only take place in some decades, due to the low baseline level of electric vehicle ownership and the slow rate of vehicle renewal. These factors are demonstrated looking, for example, at the European market statistics for the 2021/2022 ([21], [22]), which shows that, even if a small percentage increase of electric vehicles (EV) is present, out of the 11.7 million vehicles registered in the EU, the conventional ICE (Diesel and Gasoline) represent still around 85% of the total share. Considering also that the average lifespan of a gasoline car is 18 years, it is clear that the REEV and the HEV are fundamental in this gradual transition towards electrification. In the marketplace, consumer concerns over electric vehicle range and the high initial vehicle cost (largely due to the need to accommodate a sufficiently large battery for acceptable range) continues to

inhibit uptake of BEV. This is where REEV, can encourage fleet renewal, by combining the advantages of electric propulsion (while using a smaller, lighter and less expensive battery) and thermal engines, eliminating the issue of the "range anxiety". The most suitable engine for this application, due to its lightweight and compact nature, is the rotary engine. It would be designed to operate primarily at its best efficiency point, disconnected from the drivetrain, with its sole task being to recharge the battery directly, overcoming one of the disadvantages listed in 1.1.5. From what concerns the emissions, studies done in [9] and [10], demonstrated that, adopting some adjustments in the design, like zero port overlap and peripheral ports, the engine of the Mazda RX-8 (13-MSP RENESIS), despite being homologated as Euro 4, could meet the Euro 5 and 6 emission limits. It has been proved that, if these adjustments are applied on a small Wankel engine, namely the AIE 225CS [23], the advantages of lightness and smoothness of the Wankel engine can be fully exploited for REEVs. The AIE 225CS is an engine intended for UAVs purposes; as a matter of fact this is the second field where Wankel engine can achieve great success, because size, weight and vibrations are critical aspects. The power range of the current rotary UAV engines around the world varies between 20 and 90 hp, and they are usually single rotor, with a displacement smaller than 500cc. The demand for UAVs had a dramatic increase around the world [8] in the last decade and is expected to increase in the coming years. The last trend in the design of the Wankel engines for UAVs is to move towards the multi-fuel capabilities and heavy-fuels operation (e.g. kerosene, JP-8), due to fire safety regulations and availability (e.g., at smaller airports or on ships). Existing RE propulsion units are mainly designed to run on high-volatility fuels like aviation gasoline (AvGas) or regular gasoline. Actually, this tendency started also in the early 1980s, due to uncertainty over aviation gasoline prices. John Deere researched multi-fuel-capable rotary engines, and the design was proposed for several U.S. Marine combat vehicles in the late 1980s [24]. In 1986, John Deere and NASA demonstrated an SCRE with brake-specific fuel consumption (brake specific fuel consumption (BSFC)) of 310 g/kWh at 160 hp at 8000 rpm. They set a BSFC goal of 213 g/kWh by using turbo-compounding, adiabatic components, a lightweight rotor, and reduced friction [25]. Their experimentation showed that a lean mixture could lower the BSFC. However, to burn heavy fuels in combustion engines typically CI is applied; but such combustion processes are poorly implementable in REs due to their geometry: High CR, which are necessary to reach the required temperatures to ignite heavy fuels, cannot be realized or result in unfavorable high wall area and combustion chamber shapes [4]. Together with the already mentioned studies of John Deere, the only concept of CI that went beyond patented application is the Rolls Royce Diesel RE that uses one rotor just for pre-compression [26]; other designs uses external volumes to increase the CR, heat the fuel above the ignition

temperature before injecting it or use glow plugs and direct injection (DI). However, many studies demonstrated that running on heavy fuels is possible also for for spark ignition (SI). As a matter of fact, the 225 CS shows the capability to run with JP-8 jet fuel, but only after being started with AvGas [27], because cold start is a great challenge due to the bad vaporization of these fuels, that needs higher temperatures. One of the most recent studies related to the topic is the one made by the Institute for Powertrains and Automotive Technologies, together with Austro Engine, that investigates new methods to enable heavy-fuel operation on SI RE, in a public research project funded by the Austrian Research Promotion Agency (FFG). A single rotor RE, named AE R1, with 294 cc displacement and a CR of 8.7, was used to demonstrate that sufficient performance and output can be achieved on kerosene at full-load and test its cold start capabilities. The engine was equipped with an air-assisted injection system in the intake manifold instead of the standard low-pressure fuel injector to improve mixture formation and vaporization. This modification has been found to be effective in reducing wall-film buildup in the intake manifold during kerosene operation, as reported in [3]. As a result, the engine is now capable of cold starts down to -25 degrees Celsius. Operating on heavy fuels still presents some challenges that need to be addressed.

2 | CFD Basics

2.1. OpenFOAM introduction

OpenFOAM (Open Field Operation and Manipulation) is a free and open-source computational fluid dynamics (CFD) software package widely used in academic and industrial environments. OpenFOAM is written in C++ and provides a comprehensive set of tools for simulating and analyzing fluid flows. This tools are mainly executable, known as **applications**, divided in two categories: **solvers** and **utilities**. Solvers are designed to overcome specific continuum mechanics problem; and utilities are designed to perform tasks that involve data manipulation. The software is designed around a set of core libraries that provide a framework for solving partial differential equations and performing other numerical operations. OpenFOAM allows users to create and manipulate a wide range of simulation cases, from simple laminar flows to complex turbulent flows with chemical reactions and multiphase flows. In this section, a basic overview of the OpenFOAM and Lib-ICE structure is given, together with the general setup of an OpenFOAM case.

2.1.1. OpenFOAM environment

The organization of OpenFOAM follows a specific directory structure, with each case consisting of several subdirectories that contain the input files and output data for the simulation. The OpenFOAM enviroment is usually organized as follows:

```
OpenFOAM-8.x/
├── applications/
│   ├── solvers/
│   │   ├── basic/
│   │   ├── compressible/
│   │   ├── combustion/
│   │   └── .../
│   ├── test/
│   └── utilities/
│       ├── mesh/
│       ├── preProcessing/
│       ├── postProcessing/
│       └── .../
├── bin/
├── doc/
├── etc/
├── platforms/
├── src/
│   ├── finiteVolume/
│   ├── TransportModels/
│   └── .../
├── test/
├── tutorials/
└── wmake/
```

Where:

- **Applications (`$FOAM_APP`):** Contains the source files of the executable. The folder is divided in three subfolders:
  - ▷ **solvers (`$FOAM_SOL`):** source code for the distributed solvers, classified depending on the purpose they were developed for: **basic**, **combustion**, **compressible**, **discreteMethods**, **DNS**, **electromagnetics**, **financial**, **heatTransfer**, **incompressible**, **lagrangian**, **multiphase**, **stressAnalysis**
  - ▷ `test (`$FOAM_SOL`)`: contains source code that tests and shows examples of the usage of some of the OpenFOAM classes.
  - ▷ `utilities (`$FOAM_UTIL`)`: source code for the distributed solvers, classified depending on the purpose they were developed for: **mesh**, **miscellaneous**, **parallelProcessing**, **postProcessing**, **preProcessing**, **surface**, **thermophysical**. In the `mesh` folder are located important utilities used for mesh generation, like `blockMesh` and `snappyHexMesh`, and utilities for mesh manipulation and quality control, like `mergeMesh` and `checkMesh` and so on.

- **Bin:** Contains additional scripts.
- **Doc:** Contains useful documentation.
- **Platforms:** Contains binaries of executables and libraries.
- **Src:** This directory contains the source code for all the libraries. It is divided in different subdirectories each of them can contain several libraries. The most relevant are:
  - **finiteVolume** This library provides all the classes needed for the finite volume discretization, such as mesh, divergence, laplacian, gradient discretization operators, matrix solvers, and boundary conditions.
  - **OpenFOAM** This library includes the definitions of the containers used for the operations, the field definitions, the declaration of the mesh and of all the mesh features such as zones and sets.
  - **TransportModels** Contains several turbulence models.
  - **mesh** Classes for automatic mesh generation.
  - **dynamicMesh** Algorithms for moving meshes (motion, topology change, ...).
  - **dynamicFvMesh** Moving mesh classes (mixer, piston, ...).
  - **postProcessing** Treatment of simulation data (probes, samples, forces,...).
  - **parallel** Performs simulations in parallel, domain decomposition, ...
  - **functionObjects** Functions for analyzing results of simulations.
- **Test:** Applications under specific testing.
- **Tutorials:** Contains a set of tutorials for different physical problems.
- **Wmake** Scripts and rules for compilation of OpenFOAM.

2.1.2. Lib-ICE

Lib-ICE (ICE stands for internal combustion engine) is a collection of libraries and solvers for simulating IC engines using OpenFOAM® technology and developed by the ICE Group of Politecnico of Milano. The development of this library is made possible by the open-source nature of OpenFOAM, which enables users to access and modify the source code to meet their specific needs. This aspect of OpenFOAM's design ensures that the software remains flexible and adaptable to the requirements of various computational fluid dynamics (CFD) applications. The LibICE files, similarly to the main OpenFOAM folder,

are organized into the following directories:

```
Lib-ICE-8/
├── applications/
│   ├── bin/
│   ├── solvers/
│   │   ├── coldFlow/
│   │   ├── combustion/
│   │   ├── compressible/
│   │   ├── Diesel/
│   │   └── .../
│   ├── utilities /
│   │   ├── mesh/
│   │   ├── postProcessing/
│   │   ├── preProcessing/
│   │   ├── thermophysical/
│   │   └── .../
│   └── test/
├── etc/
├── lib/
└── src/
```

Where:

- **applications**: This directory contains the source files and compiled executables for the various solvers and utilities. For this thesis work, new utilities were created for the mesh generation (see 3.2), and they have been located in this folder. More precisely, the utilities will be located in the `mesh` and in the `preProcessing` folders. In addition, for the last part of the work a coldFlow solver was used, namely the `coldEngineDymFoam`, which is located in the `coldFlow` folder. This solver is a compressible flow solver in engine meshes (deformation ad-or topological changes) with spray.
- **etc**: Contains the configuration files for LibICE.
- **lib**: This directory contains the compiled libraries in the form of share-objects (in ".so" format).
- **src**: Contains the source files of the libraries, which include objects, functions, and other related code.

2.1.3. Case organization

The organization of an OpenFOAM case follows a specific directory structure, with each case consisting of several subdirectories that contain the input files and output data for the simulation. Generally, as it is shown in picture, the OpenFOAM case is determined by the files and directories in the `constant`, `system`, and `0` (the starting point of the simulation) folders.

```
Case folder/
├── constant/
│   ├── polyMesh/
│   ├── thermophysicalProperties/
│   ├── momentumTransport/
│   ├── engineGeometry/
│   ├── g/
│   ├── dynamicMeshDict/
│   └── .../
├── system/
│   ├── blockMeshDict/
│   ├── controlDict/
│   ├── createBafflesDict/
│   ├── decomposeParDict/
│   ├── fvSchemes/
│   ├── fvSolution/
│   ├── setFieldsDict/
│   └── topoSetDict/
└── 0/
    ├── CO2/
    ├── p/
    ├── T/
    ├── U/
    └── .../
```

The `system` folder contains essential dictionaries for the numerical schemes used in OpenFOAM.

The main input file for an OpenFOAM case is the `system/controlDict` file, which stores the primary control parameters for the simulation, such as the time step and the start and end points of the simulation. Other crucial files in the `system` folder include the `fvSchemes` and `fvSolution` files, defining the numerical schemes employed for solving the equations (e.g. the time stepping scheme) and the solver settings, respectively.

Furthermore, the `system` folder also houses dictionaries for mesh generation, specifically the *blockMeshDict* and *createBafflesDict* files.

The `constant` folder contains subdirectories that define the constant properties of the simulation, which include fluid properties, turbulence modeling, engine geometry, and mesh data. Within the `constant/polyMesh` directory, files describing the mesh properties—such as vertices, edges, and faces of the mesh—are stored. Fluid properties are specified in files like *thermophysicalProperties*. The *momentumTransport* file is related to turbulence modeling and provides details on the chosen turbulence model and its parameters. The *dynamicMeshDict* file defines the settings for dynamic meshing, while the *engineGeometry* file contains information about the engine's geometric properties. Lastly, the *combustionProperties* file provides details on the combustion model and related parameters.

The `0` folder contains the initial and boundary conditions (fields) of the simulation, which are defined in files such as U, p, and T. These outline the initial velocity, pressure, and temperature fields for the simulation. Boundary conditions are the interface between the fluid domain and the external environment, and play a critical role in determining the behavior of the flow. OpenFOAM provides a range of boundary conditions for different types of flow, including fixed values, prescribed gradients, and dynamic boundary conditions that can adapt to changes in the flow. Understanding how to set up the boundary conditions correctly is essential for obtaining accurate and meaningful simulation results.

2.2. Fluid-dynamic equations

Computational fluid dynamics (CFD) method is based on the formulation of conservation laws, namely the conservation of mass (continuity equation), the conservation of momentum (Newton Law) and the conservation of energy. These three non linear partial differential equations can completely determine the behavior of a fluid-mechanic system without the need of any additional dynamic law and, when applied to a viscous flow, this set of equations is known as the *Navier-Stokes equations*. To further characterize the system additional equations can be used, for example the equation of state of the working fluid, or the chemical species conservation for combusting cases. These equations are non linear and very complex and for this reason. Unless they are simplified with strong assumptions (e.g. non viscous flow, 1-D cases, steady-state), they need to be solved numerically. Before introducing the equations of the previously mentioned laws, it is useful to present the general conservation equation for a generic quantity[1] ϕ, which can

[1]The conservation law can be written in a similar way also for a generic vector $\vec{\phi}$, but this case is not reported here for the sake of conciseness.

be written in its differential form as:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \vec{U}) = \nabla \cdot (\Gamma \nabla \phi) + \vec{Q_V} + \nabla \cdot \vec{Q_S} \tag{2.1}$$

Where ϕ is the generic scalar quantity, Γ is the diffusion coefficient, Q_V is the contribution of the volumetric sources and Q_S the contribution of the surface sources. This equation states that the variation of the total amount of a quantity inside a given domain is equal to the balance between the amount of that quantity entering and leaving the considered domain, plus the contribution from eventual sources generating that quantity. In equation (2.1) the contribution of the flux has been already split in its two components. Specifically:

- **Convective flux**

$$\vec{F_c} = \nabla \cdot (\phi \vec{U}) \tag{2.2}$$

  It describes the passive transport of the conserved variable by the flow and it is proportional to the velocity of this one. It appears as a first order partial derivative and represents a non-linear term, as the velocity field depends on transported variables.

- **Diffusive flux**

$$\vec{F_d} = \nabla \cdot (\Gamma \nabla \phi) \tag{2.3}$$

  Diffusion appears as a second order partial derivative term, as a matter of fact the divergence of a gradient leads to a Laplacian term:

$$\nabla \cdot (\nabla \phi) = \Delta \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \tag{2.4}$$

  This means that the diffusion term physically represents an isotropic diffusion phenomena if the constant Γ is not varying in space.

2.2.1. Continuity equation

The continuity equation, or mass conservation, is obtained by applying the conservation to the quantity $\phi = \rho$:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{\mathbf{U}}) = 0 \tag{2.5}$$

This equation is valid for an *unsteady incompressible flow*. As it can be seen mass does not diffuse and there are no source terms, however it should be noted that in case of spraying of particles an additional source term $\dot{S_m}$ will be present. This source term represents

the mass added to or removed from the fluid due to the presence of spray particles and is typically modeled using empirical correlations based on experimental data or theoretical models. The specific form of the $\dot{S_m}$ source term depends on the particular spray model being used and the physical processes being considered, such as droplet breakup, collision, evaporation, and coalescence.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{\mathbf{U}}) = \dot{S_m} \tag{2.6}$$

2.2.2. Momentum equation

The momentum equation (Newton law) is obtained for the conservation of the vectorial quantity $\phi = \rho\vec{\mathbf{U}}$:

$$\frac{\partial (\rho \vec{U})}{\partial t} + \nabla \cdot (\rho \vec{U} \vec{U}) - \nabla \cdot \overline{\overline{\sigma}} = \rho f_e \tag{2.7}$$

Also in this case there is no diffusive contribution, while the source terms are introduced from external volume forces f_e (e.g. gravity or applied forces) and from internal forces $\overline{\overline{\sigma}}$ (stress). The internal forces cancel out inside the control volume, contrarily to what happens on the volume surface, where they do not have any counterpart. Internal forces cause deformation of the fluid and depends on the position and orientation of the surface they acts on. They are represented as a tensor, which, for a Newtonian fluid, has the following expression:

$$\overline{\overline{\sigma}} = -p\overline{\overline{I}} + \overline{\overline{\tau}} \tag{2.8}$$

Where $-p\overline{\overline{I}}$ is the isotropic pressure component, meaning the contribution of the pressure normal to every surface of the volume, while $\overline{\overline{\tau}}$ is the shear stress tensor, representing the internal friction force between fluid layers (Newton's law of viscosity). The ij-th element of the tensor is:

$$\tau_{ij} = \left[\mu \frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j} - \frac{2}{3} \nabla \cdot U \delta_{ij} \right] \tag{2.9}$$

Finally, as it was done for the mass conservation law, in presence of spray, the source term $\dot{S_m}$ must be added to the equation:

$$\frac{\partial (\rho \vec{U})}{\partial t} + \nabla \cdot (\rho \vec{U} \vec{U}) = -\nabla p + \nabla \cdot \overline{\overline{\tau}} + f_e + \dot{S_m} \tag{2.10}$$

2.2.3. Energy equation

The energy conservation equation ($\phi = \rho E$) is obtained by applying the conservation to the total energy, defined as the sum of fluid internal energy plus its kinetic energy per

unit mass:

$$E = u + \frac{1}{2}U^2 \tag{2.11}$$

The energy conservation law is the following:

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\rho E \vec{U}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\overline{\overline{\sigma}} \cdot \vec{U}) + \rho \vec{f_e} \cdot \vec{U} + q_H \tag{2.12}$$

It can be noted that, in this case, both type of fluxes appear: the convective flux is associated with the transport of mass, while the diffusive flux is associated to the thermal conductivity of the fluid (Fourier's law). The volume source terms $Q_V = \rho \vec{f_e} \cdot \vec{U} + q_H$ are related to the work of the volume forces f_e and heat sources (e.g. reactions), while the surface sources $\vec{Q_S} = \overline{\overline{\sigma}} \cdot \vec{U}$ are related to the the work done on the fluid by the internal shear stress acting on the surface of control volume. Again, if the spray is modeled there will be an additional term $\dot{Q_s}pray$ within the sources.

2.2.4. Constitutive laws

A total of five differential equations are necessary to describe the fluid motion: mass conservation equation, 3 momentum conservation equations (for x, y and z components) and energy equations. In addition to the previous set of equations, to completely characterize the system, it is necessary to introduce the constitutive laws of the fluid, which, for Newtonian fluids and compressible flow, are:

- **Equation of state**

$$\frac{p}{\rho} = nRT \tag{2.13}$$

  Where R is the universal gas constant, and n the number of moles inside the control volume.

- **Internal energy equation (or enthalpy)**

  Define the internal energy or the enthalpy as a function of pressure and temperature: $u = u(p, T)$ or $h = h(p, T)$

- **Sutherland's law for viscosity**

$$\mu = \frac{1.47 T^{\frac{3}{2}}}{T + 110} 10^{-6} \tag{2.14}$$

  This law relates the dynamic viscosity (represented by the symbol μ) of a gas to its temperature.

To summarize, the complete set of equations necessary to characterize the system is hereby

reported:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0 \\ \dfrac{\partial (\rho \vec{U})}{\partial t} + \nabla \cdot (\rho \vec{U} \vec{U}) + p\bar{\bar{I}} - \nabla \cdot \bar{\bar{\tau}} = \rho \vec{f_e} \\ \dfrac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho H \vec{U}) - k \nabla T - \bar{\bar{\tau}} \cdot \vec{U} = \rho \vec{f_e} \cdot \vec{U} + q_H \\ \dfrac{p}{\rho} = nRT \\ \mu = \dfrac{1.47 T^{\frac{3}{2}}}{T + 110} 10^{-6} \end{cases} \tag{2.15}$$

It's worth mentioning in case of turbulence modeling, the system will acquire new unknowns and it will need additional equations.

2.3. Finite Volume method

The purpose of the finite volume method is to transform one or more partial differential equations into a corresponding system of algebraic equations. The solution of this system produces a set of values corresponding to the solution of the original equations at some pre-determined locations in space and time. The discretization process can be divided into two steps:

Discretization of the solution domain: it produces a numerical description of the computational domain, including the positions of points in which the solution is sought and the description of the boundary. The space is divided into a finite number of discrete regions, called control volumes or cells. For transient simulations, the time interval is also split into a finite number of time-steps.

Equation discretization: it gives an appropriate transformation of terms of governing equations into algebraic expressions.

2.3.1. Solution domain discretization

In the Finite Volume (FV) approximation, the computation domain is divided into small control volumes (CVs) using a grid that defines the boundaries of each CV.

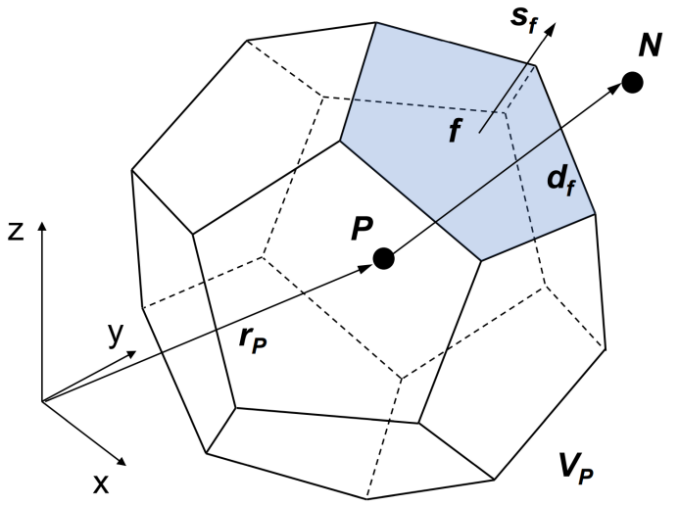


Figure 2.1: Representation of a general polyhedron cell

The cell faces in the mesh can be divided into two groups:

1. Internal faces (between two control volumes)
2. Boundary faces, which coincide with the boundaries of the domain.

Looking at Figure 2.1 it is possible to observe all the characteristics of a single element. The face area vector $\mathbf{S}_f$ points outwards from the cell, it is normal to the face, and has the magnitude equal to the area of the face. If two cells are adjacent and share a face one of the two is called "owner of the face" and is stored in the "owner" array, while the second one will be called "neighbour" and stored in the "neighbour" array. Instead, for boundary faces like S_f, the correspondent face vectors will point outwards from the computational domain. For the colored face in the Figure 2.1, the owner and neighbor cell centers are marked with P and N. They are called centroids and can be defined as follows:

$$\int_V (\vec{x} - \vec{x_P}) dV = 0 \tag{2.16}$$

This implies that the centroid of the volume V is located at the point x_P where the first moment of the volume with respect to the coordinate x is zero, it means that the centroid of V is located at the balance point of V along the x-axis. The centroids **P** and **N** are important because they are the points where the the quantities of interest will be defined and calculated. Also the internal face f will have its own centroid, and it can be defined

as:

$$\int_S (\vec{x} - \vec{x_f}) dV = 0 \tag{2.17}$$

In OpenFOAM the information about the mesh are stored in the `constant/polyMesh` folder, specifically in the files `points`, `faces owner`, `neighbour`, `polyMesh`, while the input dictionary for the grid definition `blockMeshDict` is located in the `system` folder.

2.3.2. Equations discretization

In order to reduce the problem to a system of linear algebraic equations the discretization of the transport equation is needed. The starting point for the discretization is the integral formultion of the conservation law (2.1), that grouping the terms becomes:

$$\int_V \frac{\partial \phi}{\partial t} dV + \int_S \phi \vec{U} \cdot \vec{n} dS = \int_S \Gamma \nabla \phi \cdot \vec{n} dS + \int_V q_\phi dV \tag{2.18}$$

The integral conservation equation applies to each Control Volume (CV), namely cell, as well as to the entire solution domain, meaning that that global conservation is naturally embedded in the FV method. As a matter of fact, summing up all the equations for all the CVs it is possible to obtain the global conservation equation, because the surface integrals over the inner faces cancel out. To obtain an algebraic equation for a particular CV, the surface and volume integrals over the cells must be approximated. Therefore, the main step in FV method is how to approximate these integrals.

- **Approximation of volume integral:** an accuracy of the second order can be obtained by replacing the volume integral by the mean value of the integrand (approximated by the cell center value) and the CV volume.

$$\begin{aligned} \int_V \phi dV &= \int_V \phi_P dV + \int_V (\vec{x} - \vec{x}_P) \cdot (\nabla \phi)_P dV + \int_V O((\vec{x} - \vec{x}_P)^2 \nabla \nabla \phi) dV \\ &= \phi_P \int_V dV + \left[\int_V (\vec{x} - \vec{x}_P) \right] \cdot (\nabla \phi)_P dV + \int_V O((\vec{x} - \vec{x}_P)^2 \nabla \nabla \phi) dV \\ &= \phi_P V_P + O(\Delta x^2) \end{aligned} \tag{2.19}$$

  This approximation takes the name of "midpoint rule".

- **Approximation of surface integral:** to calculate this integral, the integrand should

be known everywhere on the surface S. However, this is not the case, since the quantity ϕ is known only in the centroids. An approximation must be introduced, and this is done in two steps: firstly, the integrand is approximated in terms of variable values at one or more locations on the cell face, then the cell-face values are approximated in terms of cell center values. Note that the approximation involves interpolating values from the cell centers to the cell faces, and then integrating over the faces using the interpolated values. This process is known as flux interpolation, and it is a critical step in the finite volume method.

$$\begin{aligned}\int_S \phi dS &= \phi_f \int_S dS + \left[\int_S (\vec{x} - \vec{x}_f) dS\right] \cdot (\nabla\phi)_f + \int_S O((\vec{x} - \vec{x}_f)^2 \nabla\nabla\phi) dS \\ &= \int_S \phi dS = \phi_f S + O\left(\Delta x^2\right)\end{aligned} \tag{2.20}$$

Since ϕ_f is not known at the face center it has to be obtained by interpolation. To preserve second order accuracy ϕ_f must be computed with at least second order accuracy.

Convection term discretization

The discretization of the convection term is performed using the Gauss theorem:

$$\int_V \nabla(\phi\vec{U}) dV = \int_S \phi\vec{U} \cdot \vec{n} dS \quad = \sum_f (\phi\vec{U})_f \cdot \vec{S} \; = \sum_f \vec{S} \cdot \vec{U}_f \phi_f \quad = \sum_f S U_n \phi_f \tag{2.21}$$

The value of ϕ_f at the face midpoint can be determined following different methods, but it is important to remember that the approach used to determine the value of ϕ_f at the face midpoint will affect the accuracy of the discretization. In order to obtain consistent results, the choice of the proper convection scheme is fundamental in OpenFOAM. These are set up in the `fvSchemes` file under the `divSchemes` subdictionary. Listing some of the most common strategies, we have:

- **Upwind** Approximates the ϕ_f with the value of the neighboring cells on the basis of the flow direction:

$$\begin{aligned}&\sum_f F_n \phi_f \quad \phi_f = \phi_p + \overbrace{(\vec{x}_f - \vec{x}_P) \cdot \left(\vec{\Delta}\phi_P\right)_P (\vec{x}_f - \vec{x}_P)^2}^{\epsilon_t} \\ &\phi_f = \begin{cases} \phi_p & \text{if } \vec{U} \cdot \vec{n} > 0 \\ \phi_N & \text{if } \vec{U} \cdot \vec{n} < 0 \end{cases}\end{aligned} \tag{2.22}$$

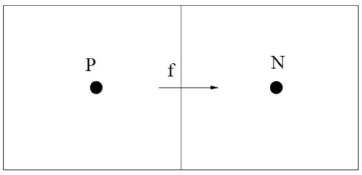


Figure 2.2: Upwind interpolation scheme.

This method is first order, since the leading term of the Taylor series expansion is proportional to Δx and it will also lead to a diffusive term, since the flux of a gradient will be present. Despite the introduction of numerical diffusivity, this approach turns out to be always bounded. By substituting (2.22) into (2.21), it follows:

$$\int_V \nabla(\phi\vec{U})dV = \int_S \phi\vec{U}\cdot\vec{n}dS = \sum_f F_n\phi_f = \sum_f F_n\phi_P + \sum_f F_n\left(\vec{x}_f - \vec{x}_P\right)\cdot(\nabla\phi)_P \tag{2.23}$$

where it can be seen from the second term that the leading term of the truncation error is proportional to a diffusive flux and it will introduce a source of artificial viscosity with the consequent risk of error. Also, this implies that a very refined mesh will be required to achieve high accuracy and in case of obliquely oriented flow with respect to the mesh there will be diffusion also in the direction normal to the flow.

- **Linear** The face center value is linearly interpolated between the two nearest cell centers:

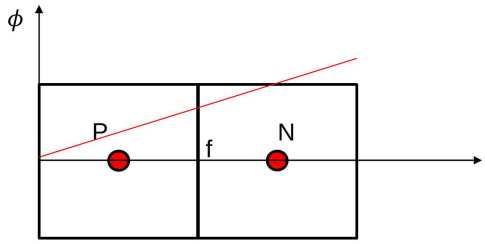


Figure 2.3: Linear interpolation scheme.

$$\begin{aligned} \phi_f &= \lambda\phi_N + (1-\lambda)\phi_P \\ \lambda &= \frac{x_f - x_P}{x_N - x_P} \end{aligned} \tag{2.24}$$

Expanding ϕ_N around point x_P to eliminate the first derivative:

$$\phi_f = \lambda\phi_N + (1-\lambda)\phi_P - \frac{(x_f - x_P)\,(x_N - x_f)}{2}\frac{\partial^2\phi}{\partial x^2} + H \tag{2.25}$$

In this case, the leading term of the truncation error is of second order, so it goes to zero quicker than the upwind case. This method is highly unstable (unconditionally unstable) and produces nonphysical oscillations in the solution, which are amplified by sharp gradients, despite there is not a diffusion term introducing artificial viscosity.

- **Linear upwind** The linear upwind differencing (LUD) scheme involves two upstream values for the calculation of the face center value expression:

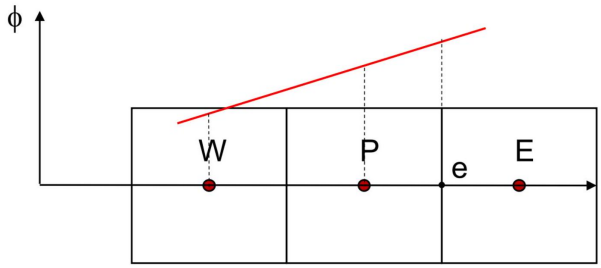


Figure 2.4: Linear upwind interpolation scheme.

$$\phi_e = \phi_P + \frac{\phi_P - \phi_W}{\delta x}\frac{\delta x}{2} \tag{2.26}$$

This can be thought of as a second-order extension of the original upwind estimate.

Gradient term discretization

The discretization of the diffusion term is performed using this derivation of the Gauss theorem:

$$\begin{aligned}\int_V \nabla\phi dV &= \int_S \phi\vec{n}dS \\ &= \sum_f \phi_f\vec{S}\end{aligned} \tag{2.27}$$

ϕ_f is evaluated resorting to linear interpolation between the two adjacent nodes leading to a second order accurate discretization.

Due to the assumed linear variation of the property ϕ, the gradient can be considered

constant and for this:

$$\begin{aligned}\int_V \nabla \cdot (\Gamma \nabla \phi) dV &= \int_S \Gamma \nabla \phi \cdot \vec{n} dS \\ &= \sum_f (\Gamma \nabla \phi)_f \cdot \vec{S} \\ &= \sum_f \Gamma_f (\vec{S} \cdot \nabla \phi)_f\end{aligned} \tag{2.28}$$

The term Γ_f is interpolated onto the face, while the term $(\vec{S} \cdot \nabla \phi)_f$ depends on the type of mesh considered.

$$(\vec{S} \cdot \nabla \phi)_f = \left|\vec{S}_f\right| \frac{\phi_N - \phi_P}{|\vec{d}|} \tag{2.29}$$

This formulation allows to achieve second order accuracy on uniform and orthogonal meshes. The definition of the proper diffusion scheme in OpenFOAM is done, as for the other discretized terms, in the `fvSchemes`, but under the `laplacianSchemes` subdictionary.

Mesh quality parameters

The discretization previously presented, which allows the switch from continuous to discrete domain, can introduce some errors. For what concerns the mesh, these errors can be quantified thanks to specific parameters, namely **skewness**, **mesh non-orthogonality** and **flow orientation** of the grid. These characteristics play a crucial role in the accuracy and stability of numerical simulations, as well as the convergence of the solution.

- **Skewness:** Skewness is a measure of the deviation of a cell's shape from an ideal or regular shape, e.g. an hexahedron cell in 3D. High skewness can lead to reduced accuracy and slower convergence of the numerical solution, as the interpolation and flux calculations in highly skewed cells may introduce errors. In general, it is recommended to maintain low skewness. For the wankel engine case these is not an easy task, due to its restrictive geometry and cells distortion.

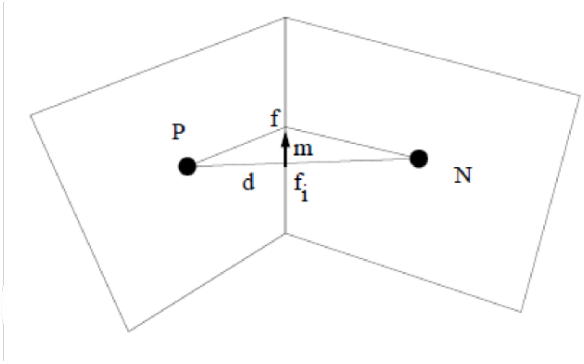


Figure 2.5: Skewness representation.

Index of skewness is given by:

$$\psi = \frac{|\vec{m}|}{|\vec{d}|} \tag{2.30}$$

$$\begin{aligned} \phi_f &= \phi_i + (\vec{x}_f - \vec{x}_i) \cdot \nabla\phi = \phi_i + \vec{m} \cdot \nabla\phi \\ \phi_i &= \lambda\phi_P + (1-\lambda)\phi_N \end{aligned} \tag{2.31}$$

Consequently, it can be demonstrated that the truncation error for convection and gradient terms will be function of the mesh spacing and of the skewness index.

$$\epsilon_t = \psi \Delta x^m \frac{\partial^{m+1}\phi}{\partial x^{m+1}}\bigg|_i \tag{2.32}$$

- **Mesh Non-Orthogonality:** Mesh non-orthogonality refers to the deviation of the grid lines from being orthogonal or perpendicular to each other. In an ideal orthogonal mesh, the grid lines intersect with 90° angles. Non-orthogonal meshes can be employed to accommodate complex geometries and boundary conditions in CFD simulations, like in the Wankel engine case. However, non-orthogonal meshes can introduce numerical errors and stability issues, as the discretization and interpolation schemes are more complex compared to orthogonal meshes. It is important to carefully balance non-orthogonality with the need for accurate and stable simulations.

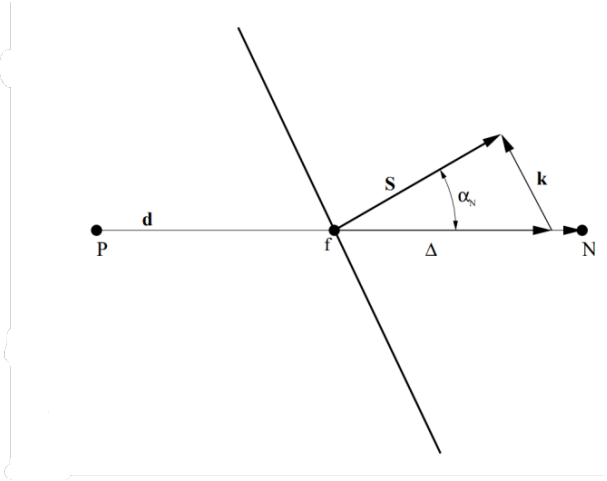


Figure 2.6: Non orthogonality vectors.

As a matter of fact, in case of non orthogonal mesh, since vectors $\vec{d}$ and $\vec{S}$ are no longer parallel, the equation (2.32) will be no longer valid and the approximation of $(\vec{S} \cdot \vec{\nabla}\phi)_f$ is done with:

$$(\vec{S} \cdot \nabla\phi)_f = \frac{\left|\vec{S}_f\right|}{\cos\alpha_N} \frac{\phi_N - \phi_P}{|\vec{d}|} - \vec{k}(\nabla\phi)_f \tag{2.33}$$

This means that with non orthogonality the accuracy is penalized and the correction may introduce unboundedness. $\vec{k}$ is the correction to compensate the non orthogonality, while the first term is related to the orthogonal contribution. There are three different ways to decompose the vector $\vec{S}$ into $\vec{k}$ and $\vec{\Delta}$ (minimum correction, orthogonal correction, over-relaxed [28], [29]), but for sake of conciseness they are not reported here.

- **Flow-Oriented Grid:** A flow-oriented grid is a mesh design strategy where the grid lines are aligned with the primary flow direction or the expected streamline patterns. This approach helps in reducing the numerical diffusion introduced by the discretization of the governing equations and improves the accuracy of the solution. Flow-oriented grids are particularly useful in simulations with strong anisotropic features, such as boundary layers, shear layers, and highly swirling flows. By aligning the grid lines with the flow, the interpolation errors can be minimized, and the convergence rate of the solution can be improved. However, also in this case it is quite impossible to obtain this alignment on a Wankel engine simulations, due to the turbulence generated.

For what concerns the OpenFOAM software, a useful tool to evaluate these parameters

is the `checkMesh` utility, which is able to give a good overall evaluation of the mesh at each time step.

Time derivative term discretization

For time varying problems the discretization of time derivative $\int_V \frac{\partial \phi}{\partial t} dV$ is also necessary. To evaluate the volume integral the midpoint rule can be applied:

$$\int_V \frac{\partial \phi}{\partial t} dV = \left.\frac{\partial \phi}{\partial t}\right|_P V_P$$

Then, it is necessary to introduce the time step Δt, which is the interval it will be adopted to march ahead in time:

$$t_{\text{new}} = t_{old} + \Delta t$$

There are two basic types of time advancement: **implicit** and **explicit** schemes. Properties of the algorithm critically depend on this choice, but both are useful under given circumstances. In the **explicit** methods ϕ_P^n (the value of ϕ_P at the new timestep) is calculated using the old time neighbor values ϕ_N^o (the apex stands for "old", meaning from the previous timestep). This method is fast and efficient, but poses Courant number limitations. If an **implicit** method is applied, instead, the ϕ_P^n will depend on the new time neighbor values ϕ_N^n. Then the temporal derivative can be discretized with different strategies:

- **Euler** This method has first-order accuracy, meaning that the error decreases linearly with the time step.

$$\frac{\partial \phi}{\partial t} = \frac{\phi^n - \phi^o}{\Delta t} \tag{2.34}$$

  It can be explicit or implicit. The Explicit Euler (Forward Euler) method uses the forward difference to approximate the time derivative at the current time level, while the Implicit Euler (Backward Euler) method uses the backward difference to approximate the time derivative at the next time level and, for this, it involves solving a non linear system. The first one is Conditionally stable and the time step should be small to avoid instability and divergence, while the second is unconditionally stable, but large timesteps may lead to poor accuracy.

- **Backward differencing** This method has second-order accuracy, meaning that the error decreases quadratically with the time step, and it is conditionally stable. The

time derivative is approximated using a second-order backward difference:

$$\frac{\partial \phi}{\partial t} = \frac{\frac{3}{2}\phi^n - 2\phi^o + \frac{1}{2}\phi^{oo}}{\Delta t} \tag{2.35}$$

- **Crank-Nicolson** The Crank-Nicolson method is a semi-implicit time discretization scheme that combines the properties of both the explicit and implicit Euler methods. It achieves this by taking the average of the derivatives at time levels o and n. This method has second-order accuracy in time and is unconditionally stable.

$$\frac{\partial \phi}{\partial t} = \frac{1}{2}\left(\frac{\phi^n - \phi^o}{\Delta t} + \frac{\phi^o - \phi^{oo}}{\Delta t}\right) \tag{2.36}$$

Simplifying the equation:

$$\frac{\partial \phi}{\partial t} = \frac{\phi^n - \phi^o}{\Delta t} - \frac{1}{2}\left(\frac{\phi^o - \phi^{oo}}{\Delta t}\right) \tag{2.37}$$

In OpenFOAM the time advancing scheme is defined in the entry `ddtSchemes` of the `fvSchemes file`.

Source term discretization

In general, q_P may be a function of space and time, of the solution itself, of other variables and can be quite complex.

$$\int_V q_\phi dV = q_P V_P \tag{2.38}$$

Typically, linearisation with respect to ϕ is performed to promote stability and boundedness.

$$q(\phi) = q_u + q_d \phi \tag{2.39}$$

where $q_d = \frac{\partial q_u}{\partial \phi}$. If $q_d < 0$ it can be used to enforce diagonal dominance of the solving matrix improving the convergence of the system.

2.4. Linear equation system

2.4.1. Matrix assembly

The choice of the numerical method to be used for the discretization (assuming that consistency is respected) establishes a relation between the value of the variable ϕ_P in a

specific cells (the unknown) and the value of the same variable at different point locations ϕ_N (neighboring cells). This creates a linear equation for every cell and a system of linear equations, if we consider the whole mesh. The solution of this system will be the solution of the simulation domain. This system can be written in matrix form, where the contribution to the matrix coefficient will depend on the numerical method, the domain discretization method and the time advancing strategy used. For each computational point, an equation is created:

$$a_P \phi_P + \sum_N a_i \phi_i = Q \tag{2.40}$$

Where N is the total number of neighbouring cells of the computational cell. The matrix is constructed so that every term related to ϕ_P adds a contribution to a_P (the diagonal term) and every time ϕ_N adds a contribution to a_N (the off-diagonal terms). All the other terms, like values known from the previous steps or any other known value, contribute to the source term Q.

The convection and diffusion terms have different effects on the diagonally dominant matrix and the boundedness of the solution. For the convection term, Upwind Differencing creates a diagonally equal matrix, while other schemes introduce negative coefficients and violate diagonal equality, potentially leading to unbounded solutions. Linear differencing on a uniform mesh further complicates the issue. For the diffusion term, a diagonally equal matrix is formed only on orthogonal meshes. On non-orthogonal meshes, the correction term introduces negative coefficients in the matrix, violating diagonal equality and modifying the boundedness of the solution. This also increases the computational effort due to a higher number of non-zero matrix coefficients. To address these issues, the diffusion term can be split into the implicit orthogonal contribution (first neighbors, diagonally equal matrix) and the non-orthogonal correction (second neighbors, added to the source term). This approach maintains a reasonable computational effort while considering the non-orthogonal correction.

This creates a **sparse** matrix (the majority of the extra-diagonal elements are zero) and moves the problem to the solution of a linear system.

$$[\mathbf{A}][\phi] = [Q]$$

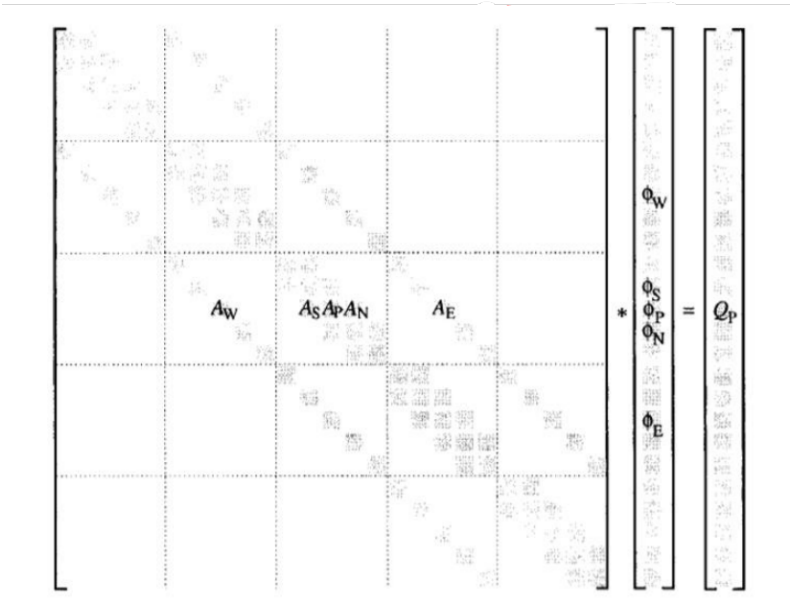


Figure 2.7: Matrix assembly.

The system created in this way is very complex an computationally demanding due to the number of equations to be solved, so the solution can be found using different methods. The two main categories of solvers are:

- **Direct methods:** give the solution of the system of algebraic equations in a finite number of arithmetic operations. To cite some: Matrix inversion, Gauss Elimination, LU decomposition. These are really computationally expensive (number of operations is proportional to the cell number to the power of 3).
- **Iterative methods:** start with an initial guess and then continue to improve the current approximation of the solution until some "solution tolerance" is met. These solvers are more economical, but they usually pose some requirements on the matrix, which usually is improved to increase its diagonally dominance, which enhance convergence. They are suitable for FV method and the only option for large problems. To cite some: Jacobi, Gauss Siedel, BCG and so on.

In OpenFOAM it is possible to choose many different solution strategies, defining them in the `fvSolution` file under the `solvers` subdictionary.

2.4.2. Boundary conditions

The computational mesh includes a series of faces which coincide with the boundaries of the physical domain under consideration. Boundary conditions are used to communicate

to the solver the information about the physics outside of the computational domain in correspondance of those faces. The boundary conditions are divided into **numerical** and the **physical** one. There are two basic types of **numerical** boundary conditions, which can be declined or combined to handle every type of boundary:

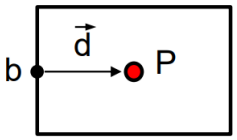


Figure 2.8: Simplified cell schematization.

- **Dirichlet (or fixed value) boundary condition** prescribes the value of the variable on the boundary.

$$\phi_b = a_b \tag{2.41}$$

- **Von Neumann boundary condition** prescribes the gradient of the variable normal to the boundary.

$$\phi_b = \phi_P + \vec{d} \cdot \nabla\phi \tag{2.42}$$

Where a_b, $\nabla\phi$ components will enter the $[Q]$ vector.
Physical boundary conditions, instead, include symmetry planes, walls, inlet and outlet conditions, adiabatic or fixed temperature boundaries for heat transfer problems and so on. Each of these conditions is associated with a set of numerical boundary conditions on each of the variables being calculated.

2.5. Turbulence modeling

Turbulence is a complex fluid flow phenomenon that occurs in a wide range of natural and engineering processes and is characterized by an irregular, chaotic, and rapidly changing motion of fluid particles. Turbulence is characterized by *fluctuations* in flow quantities that occur even in the presence of time-invariant boundary conditions, leading to a chaotic behavior. This is partly due to the *non-linearity* of the Navier-Stokes (NS) equations, which triggers turbulence for sufficiently high values of non-linearity parameters, such as the Reynolds number (Re). As a matter of fact, *turbulence* is the final state of a series of non-linear instabilities called transition. *Vorticity* is another key feature of turbulent flows, characterized by fluctuating eddies that undergo distortion and spin. These eddies vary in size, ranging from the problem's length scale to very small scales. The larger the Reynolds number, the smaller the eddy scale is. Momentum transfers from

larger to smaller eddies in an inviscid manner until viscous *dissipation* occurs at the level of the smallest eddies. Finally, turbulent flows exhibit enhanced *diffusivity* due to chaotic motion. This increased mixing results in mass, momentum, and energy diffusion rates that are much larger than molecular ones [30].
In this context, turbulence modeling aims to simulate and predict turbulent flows using mathematical models and numerical techniques. Three commonly used approaches in turbulence modeling are Reynolds-Averaged Navier-Stokes (RANS), Large Eddy Simulation (LES), and Direct Numerical Simulation (DNS).

Direct Numerical Simulation (DNS): it is the most accurate and computationally intensive approach among the three. It resolves all the turbulent scales by directly solving the Navier-Stokes equations without any modeling. Due to its high computational cost, DNS is mainly used for fundamental research and is limited to relatively simple geometries and low Reynolds number flows.

- **Reynolds-Averaged Navier-Stokes (RANS):** RANS is a widely used method for engineering applications because of its relatively low computational cost. It decomposes the flow variables into mean and fluctuating components, and then solves the time-averaged equations. RANS requires additional equations to model the turbulent stresses, known as turbulence closure models. Popular RANS models include k-epsilon and k-omega models.

Large Eddy Simulation (LES): LES is an intermediate approach between RANS and DNS in terms of computational cost and accuracy. In LES, the large-scale turbulent motions are resolved explicitly, while the smaller, unresolved scales are modeled using sub-grid scale models. This allows for more accurate and detailed flow predictions than RANS, making LES suitable for complex flow phenomena.

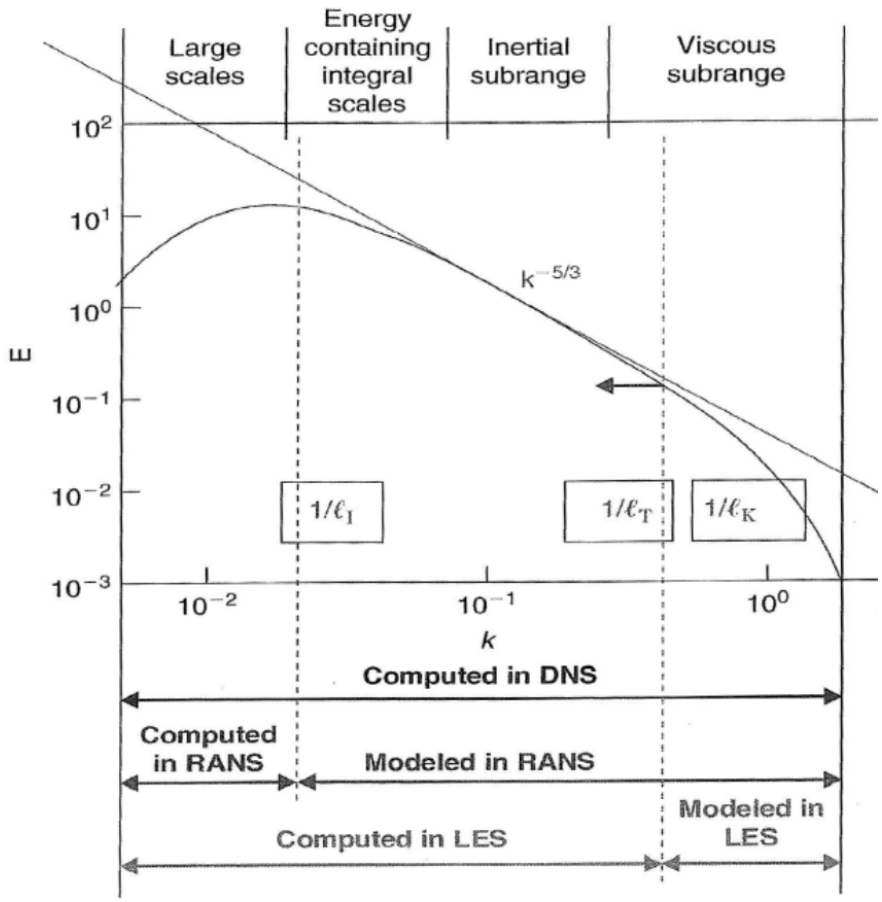


Figure 2.9: Turbulence scales.

2.5.1. RANS

As mentioned above, the RANS method is widely applied in the engineering field and it has been used also in this thesis work (in form of k-ϵ). The first step to model turbulence with RANS is to separate the local value of the variable into the mean and the fluctuation around the mean. The selection of averaging methods depends on the characteristics of turbulent flow, however, all these methods can appear in two versions, unweighted (Reynolds) and weighted (e.g. density-weighted Favre averaging).
The Reynolds averaging technique operates as follows:

$$\phi(\mathbf{x}, t) = \bar{\phi}(\mathbf{x}, t) + \phi'(\mathbf{x}, t) \tag{2.43}$$

Where $\phi'(\mathbf{x},t)$ denotes the fluctuation about the mean value, defined as:

$$\bar{\phi}(\mathbf{x},t) = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} \phi_i(\mathbf{x},t) \tag{2.44}$$

Reynolds averaging is generally applied to incompressible flows, while Favre averaging is used for compressible flows. Favre averaging compute the mean component as a density weighted average:

$$\tilde{\phi}(\mathbf{x},t) = \frac{\overline{\rho\phi}}{\bar{\rho}} \tag{2.45}$$

In both cases (Reynolds or Favre) fluctuating components have zero average.
The subsequent step is to introduce the averaged quantities inside the conservation law equations. For what concerns the mass conservation equation, applying the Reynolds averaging and taking the mean of the mass equation, the fluctuating component is eliminated and the equation is equal to the instantaneous one:

$$\overline{\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\overrightarrow{U}) + \nabla\cdot(\rho\vec{u})} = 0 \Rightarrow \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\overrightarrow{U}) = 0 \tag{2.46}$$

This does not happen for the momentum conservation and the energy conservation equations, since the averaging introduces new stresses due to the non linear term presence inside them.

$$\begin{aligned}
&\frac{\partial(\rho\vec{U})}{\partial t} + \nabla\cdot(\rho\vec{U}\vec{U}) = \rho\vec{g} - \nabla P + \mu\nabla^2\vec{U} + \frac{1}{3}\mu\nabla(\nabla\cdot\vec{U}) \Rightarrow \\
&\frac{\partial(\rho\vec{U})}{\partial t} + \overrightarrow{\nabla\cdot(\rho(\overrightarrow{U}+\vec{u})(\overrightarrow{U}+\vec{u}))} = \rho\vec{g} - \nabla\vec{P} + \mu\nabla^2\vec{U} + \frac{1}{3}\mu\nabla(\nabla\cdot\overrightarrow{U}) \Rightarrow \\
&\frac{\partial(\rho\overrightarrow{U})}{\partial t} + \nabla\cdot(\rho\overrightarrow{U}\,\overrightarrow{U}) = \rho\vec{g} - \nabla\bar{P} + \mu\nabla^2\overrightarrow{U} + \frac{1}{3}\mu\nabla(\nabla\cdot\overrightarrow{U}) - \nabla\cdot(\rho\,\overrightarrow{\vec{u}}\,\overrightarrow{\vec{u}})
\end{aligned} \tag{2.47}$$

The fluctuating component, expressed in divergence form and put on the right hand side, acts like a stress added to the viscous molecular one and is called **Reynolds stress tensor**:

$$\bar{\bar{r}} = -\rho\overline{(\vec{u}\vec{u})} = -\rho\begin{bmatrix} \overline{u_x^2} & \overline{u_x u_y} & \overline{u_x u_z} \\ \overline{u_y u_x} & \overline{u_y^2} & \overline{u_y u_z} \\ \overline{u_z u_x} & \overline{u_z u_y} & \overline{u_z^2} \end{bmatrix} \tag{2.48}$$

$$tr(\bar{\bar{r}}) = -\rho\left(\overline{u_x^2} + \overline{u_y^2} + \overline{u_z^2}\right) = -2\rho\kappa \tag{2.49}$$

The trace of this tensor is related to the turbulent kinetic energy κ. In fact, the Reynolds stresses physically represent the driving force (turbulent transport) that enhances diffusivity and mixing rate in turbulent flows. The same procedure can be applied to the energy equation, obtaining:

$$\frac{\partial(\rho c\bar{T})}{\partial t} + \nabla \cdot (\rho c\bar{T}\overrightarrow{\bar{U}}) = k\nabla^2\bar{T} - \nabla \cdot \left(\rho c\overrightarrow{T'\vec{u}}\right) \tag{2.50}$$

The non-linear convective term causes the presence of a combined fluctuating term analogous to the Reynolds stress, which is called **turbulent heat flux**. Like the Reynolds stresses, the turbulent heat flux represents the way in which the turbulent transport enhances the diffusivity with respect to the molecular one. As it can be seen these tensors introduce new unknowns creating a closure problem for the overall system, leading to the use of some models to solve it. Applying the **Boussinesq's hypothesis** of eddy viscosity the Reynolds stress tensor can be expressed as sum of an isotropic and a deviatoric anisotropic component. The elements of the tensor can be computed as:

$$-\rho\overline{u_i'u_j'} = \left[\mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) + \frac{2}{3}\rho k\delta_{ij}\right] \tag{2.51}$$

Where k is the **turbulent kinetic energy**, computed as: $k = \frac{1}{2}\left(\overline{u_x^2} + \overline{u_y^2} + \overline{u_z^2}\right)$ and μ_t is the **turbulent viscosity**. The turbulent viscosity μ_t can be evaluated in many different ways, ranging from algebraic relations and local equilibrium assumptions to the solution of transport equations. The most popular approach is to express μ_t as a function of the turbulent kinetic energy k and its dissipation rate ε (k-ε method) or the turbulent frequency ω. This leads to a "two-equation" turbulence model:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{2.52}$$

or

$$\mu_t = \rho C_\mu \frac{k}{\omega} \tag{2.53}$$

where k and ε/ω are obtained by the solution of their respective (or equivalent) transport equations. These models have advantages and drawbacks; in particular the $k-\varepsilon$ method guarantees acceptable accuracy for simple flows and it is good to model what happens far from the walls, but it is problematic near the wall region since one term of its transport equation tends to singularity. This issue is treated using particular **wall functions**. The $k-\omega$ usually require less tuning and does not need wall functions usage, but it can be less accurate and more problematic for modeling the free stream turbulence. Obviously,

simpler methodologies (1-equation models) or more complex models ($k-\omega$ SST) are available today, and also OpenFOAM allows to choose between many of them. The turbolence model in OpenFOAM 8 must be specified in the `constant` folder of the case in the `momentumTransport` file.

2.6. Solution strategies

To understand why some specific solution strategies must be adopted to solve the equations it is useful to visualize again the system of equations (conservative form) to be solved:

Mass Conservation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{2.54}$$

Momentum Equation

$$\frac{\partial (\rho \mathbf{U})}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) - \nabla \cdot \left[\mu \left(\nabla \mathbf{U} + (\nabla \mathbf{U})^T\right)\right] = \rho \mathbf{g} - \nabla \left(P + \frac{2}{3}\mu \nabla \cdot \mathbf{U}\right) \tag{2.55}$$

Energy Equation

$$\begin{aligned}\frac{\partial (\rho e)}{\partial t} + \nabla \cdot (\rho e \mathbf{U}) - \nabla \cdot (k \nabla T) = \rho \vec{g} \cdot \mathbf{U} - \nabla \cdot (P\mathbf{U}) \\ - \nabla \cdot \left(\frac{2}{3}\mu (\nabla \cdot \mathbf{U})\mathbf{U}\right) + \nabla \cdot \left[\mu \left(\nabla \mathbf{U} + (\nabla \mathbf{U})^T\right) \cdot \mathbf{U}\right] + \rho Q\end{aligned} \tag{2.56}$$

Where for both the momentum and energy equations the diffusion term is split between molecular and turbulence term.
Mass and momentum equation are strongly coupled since the rate of change of density depends on the face flux, which is proportional to the velocity. Also the energy equation is coupled with the rest of the system, since temperature influences the density trough the equation of state. With the constitutive laws ($\rho = \rho(P,T)$ and $e = e(P,T)$), the thermo-physical properties of the fluid (given by thermal conductivity $k = k(P,T)$ and dynamic viscosity $\mu = \mu(P,T)$), and the two non-linear equations coming from the turbulence modeling ($k-\epsilon$), the system in composed by 7 equations and 7 unknowns. However, since pressure doesn't have an associated equation, pressure must be calculated to solve the set of equations. In other words, some specific algorithms are needed to solve the discretized Navier-Stokes equations because their solution involves the coupling between

the momentum and continuity equations. The momentum equations determine the velocity field, while the continuity equation enforces mass conservation. The pressure field is the driving force that ensures the velocity field satisfies mass conservation. Although, the pressure does not appear explicitly in the momentum equations, creating a challenge in solving these coupled equations.
PISO, SIMPLE, and PIMPLE algorithms provide iterative methods to handle this coupling between the momentum and continuity equations. They are based on a **segregated approach** where the equations are solved sequentially rather than simultaneously. These algorithms are advantageous in terms of computational efficiency and memory requirements when compared to coupled solvers, which solve the momentum and continuity equations simultaneously in a single matrix system.
PISO (Pressure-Implicit with Splitting of Operators) is a semi-implicit method primarily used for transient incompressible flows. The main idea of the PISO algorithm is to decouple the pressure and velocity calculations in two separate steps called Corrector and Predictor.
The algorithm involves the following steps:

1. Solve the momentum equation to obtain an intermediate velocity field.
2. Apply boundary conditions to the intermediate velocity field.
3. Solve the pressure equation using the divergence of the intermediate velocity field.
4. Correct the velocity field using the pressure field.
5. Perform multiple pressure-velocity corrections (PISO loops) to improve the solution.

SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm is an iterative method mainly used for steady-state incompressible flows.
Its steps are the following:

1. Guess an initial pressure field.
2. Solve the momentum equations for the velocity field using the guessed pressure field.
3. Correct the pressure field using the continuity equation.
4. Update the velocity field using the corrected pressure field.
5. Iterate until convergence is achieved.

PIMPLE is a hybrid algorithm that combines the features of both PISO and SIMPLE methods. It is suitable for both transient and steady-state incompressible flows. The main idea of the PIMPLE algorithm is to use the PISO method for transient calculations

and the SIMPLE method for steady-state calculations, adapting the number of PISO corrector loops based on the problem requirements.
The PIMPLE algorithm works as follows:

1. Guess an initial pressure field.
2. Solve the momentum equations for the velocity field using the guessed pressure field.
3. Correct the pressure field using the continuity equation.
4. Update the velocity field using the corrected pressure field.
5. Perform multiple pressure-velocity corrections (PISO loops) if necessary, especially for transient flows.
6. Iterate until convergence is achieved for steady-state problems or advance to the next time step for transient problems.

This last algorithm is particularly useful in OpenFOAM, where it is implemented as a solver for a wide range of problems due to its versatility. Since the Wankel simulation involves compressible flows, the PIMPLE solution strategy was adopted, selecting it in the `fvSolution` dictionary .

2.7. Grid motion

In case of simulations with time-varying geometry usually it happens that the shape of the computational domain changes during the solutions, either in a prescribed manner or as part of the solution. The specific case of Wankel engine falls in the **prescribed motion** case, meaning that the grid movement is independent from the solution and it is possible to pre-define a sequence of meshes or mesh changes that will accommodate the change in the combustion chamber shape (see 3.5.2). This implies positioning the points of the mesh at each time step of the simulation and parametrically refers them to the respective CAD. A distinction can be made from the motion of the **boundary points** and the **internal points** motion, as a matter of fact, the first one can be considered as given and are prescribed by external factors (e.g. the motion of the rotor profile), while the second has the role to accomodate the boundary motion while preserving the mesh validity and quality. When the moving frame is considered, the formulation of the conservation equation will be affected because the control volume is deforming or moving with a certain velocity, leading to an additional term in the convective flux. For simplicity,

this can be demonstrated starting from a 1-D formulation of the continuity equation [31]:

$$\int_L \frac{\partial \rho}{\partial t} dx + \int_L \frac{\partial \left(\rho U_x\right)}{\partial x} dx = 0 \tag{2.57}$$

Referring to a moving boundary case:

$$\int_{x_1(t)}^{x_2(t)} \frac{\partial \rho}{\partial t} dx + \int_{x_1(t)}^{x_2(t)} \frac{\partial \left(\rho U_x\right)}{\partial x} dx = 0 \tag{2.58}$$

At this point, recalling the Leibniz's rule:

$$\int_{x_1(t)}^{x_2(t)} \frac{\partial f}{\partial t} dx = \frac{\partial}{\partial t} \int_{x_1(t)}^{x_2(t)} f(x,t) dx - f\left(x_2(t), t\right) \frac{\partial x_2}{\partial t} + f\left(x_1(t), t\right) \frac{\partial x_1}{\partial t} \tag{2.59}$$

And applying it to the continuity equation, working it out, it is possible to obtain:

$$\frac{d}{dt} \int_{x_1}^{x_2} \rho dx - \int_{x_1}^{x_2} \frac{\partial}{\partial x} \left[\rho \left(U_x - U_b\right)\right] dx \Longrightarrow \frac{d}{dt} \int_V \rho d\Omega - \int_V \nabla \cdot \rho \left(U - U_b\right) d\Omega \tag{2.60}$$

Since the terms $\frac{\partial x_i}{\partial t}$ express the velocity at which the integration boundaries are moving with, the formula can be rearranged as:

$$\frac{d}{dt} \int_{x_1}^{x_2} \rho dx - \int_{x_1}^{x_2} \frac{\partial}{\partial x} \left[\rho \left(U_x - U_b\right)\right] dx \Longrightarrow \frac{d}{dt} \int_V \rho d\Omega - \int_V \nabla \cdot \rho \left(U - U_b\right) d\Omega \tag{2.61}$$

Where it can be seen that the divergence term contains a new term, namely the **mesh flux**, that is subtracted from the variable one. This concept is a fundamental of automatic mesh motion, which has the objective to determine the motion of internal mesh points in a way that conforms to the given boundary motion while preserving mesh quality and validity. This is achieved through a combination of mathematical algorithms and numerical techniques that ensure smooth and accurate mesh deformation.
For example the equation used to describe the displacement field in this context is generally the Laplace equation:

$$\nabla \left(\gamma \nabla \mathbf{U}_b\right) = 0$$

The point velocity field, represented by $\mathbf{U}_b$, indicates the motion of the mesh points, while γ represents a scalar diffusion coefficient.
It controls the degree of smoothing in the mesh motion. A higher value results in a smoother mesh deformation, while a lower value allows for a more localized deformation.

A simple and straightforward way to describe the process of updating the positions of mesh points based on their velocity field and a given time step Δt, is given by the equation:

$$x_{\text{new}} = x_{\text{old}} + \mathbf{U}_b \Delta t \tag{2.62}$$

In OpenFOAM 8, automatic mesh motion is facilitated by a series of solvers and utilities designed to handle mesh deformation and updating during the simulation process. The mesh motion capabilities in OpenFOAM 8 are primarily implemented using a suitable solver that can handle dynamic mesh motion, such the one used for this work, i.e. `coldEngineDyMFoam` and the `constant/dynamicMeshDict` dictionary file. In the next chapter the modeling of the prescribed motion an the related mesh creation will be presented in a detailed way.

3 | Grid generation with OpenFOAM

This chapter is dedicated to the presentation of the mesh of a three-dimensional CFD model implemented in OpenFOAM® 8. After an initial presentation of the fundamental equations for the creation of its geometries and kinematics, it will be explained how the model was implemented at the code level and the operation of the various utilities. Finally, after explaining how the dictionaries should be properly set up, the consistency of the model will be shown with respect to the theoretical equations regarding the engine displacement law and the compression ratio. For this last task, the dimensions of the AIE 225CS engine and a MATLAB script were used.

3.1. Equations

The geometrical shapes of the rotor and the rotor housing are obtained by complex considerations that are extensively covered in [4]. Hereafter a brief recap of the main equations is presented. The main geometrical parameters needed to understand the following sections are listed in Table 3.1 and presented in Figure 3.1 . The equations of the shapes were reproduced in MATLAB[1] and can be seen in Figure 3.2.

[1]The demonstrative shapes are created with the specs of the AIE225CS engine, that will be presented later in section 3.5.1.

Basic Dimensions	
R	Rotor radius
e	Eccentricity
a	Parallel transfer
b	Thickness of the rotor
S_p	Minimum clearance between rotor and rotor housing
a'	Difference between a and S_p
ϕ	Oscillation angle

Table 3.1: Main Wankel dimensions

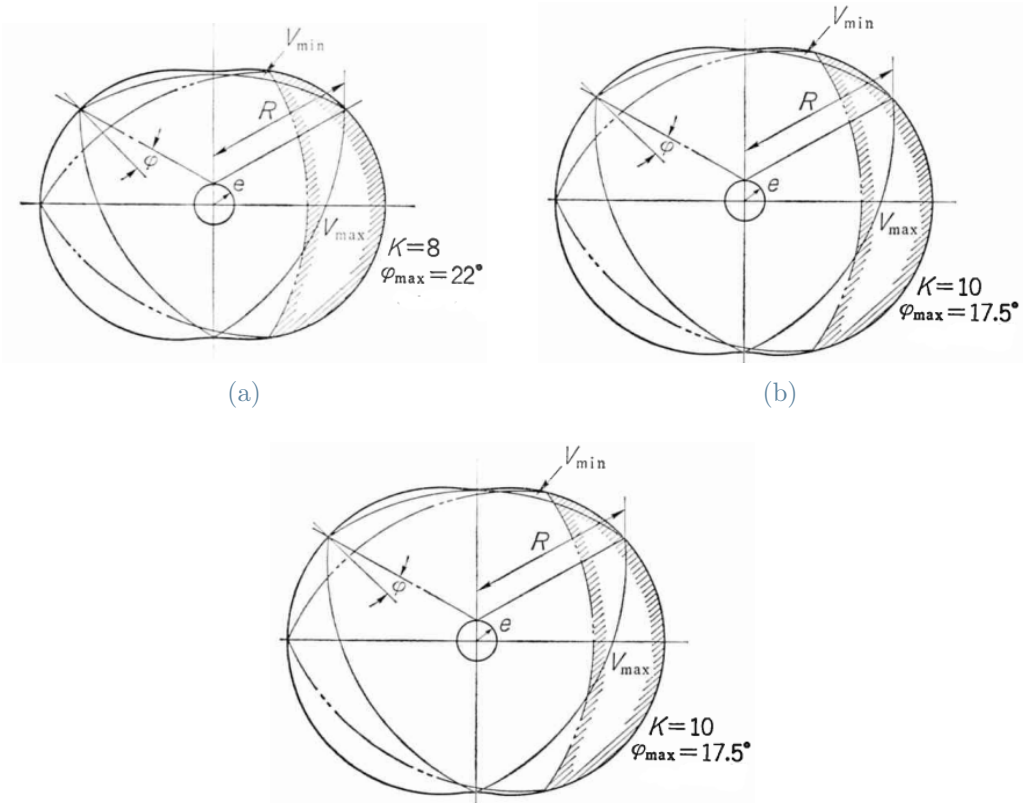


Figure 3.1: Wankel engine basic dimensions and effect of parameter K from [4]

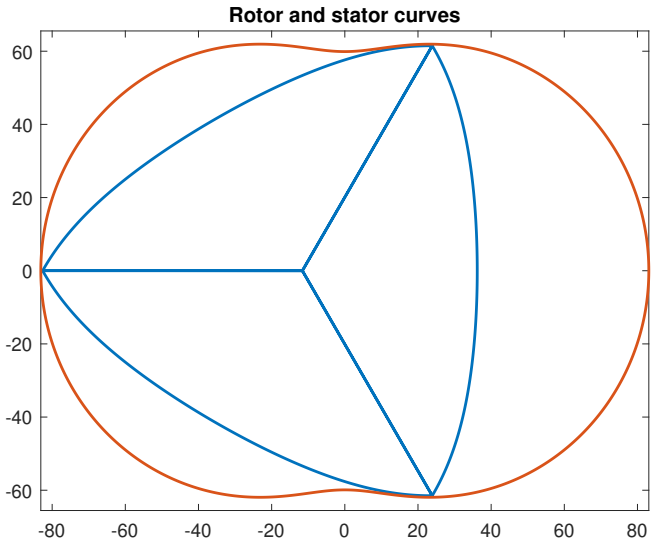


Figure 3.2: Stator and rotor curves with MATLAB.

3.1.1. Stator shape

The stator, more specifically the rotor housing (red curve in figure 3.2), is created considering the shape of a peritrochoid, a type of mathematical curve that is created by the rolling of a smaller circle with respect to a bigger one (more on [4]); however, a schematic representation of how each point of the curve is generated and their relation with the α angle (angle of the eccentric shaft) is shown in Figure 3.3.

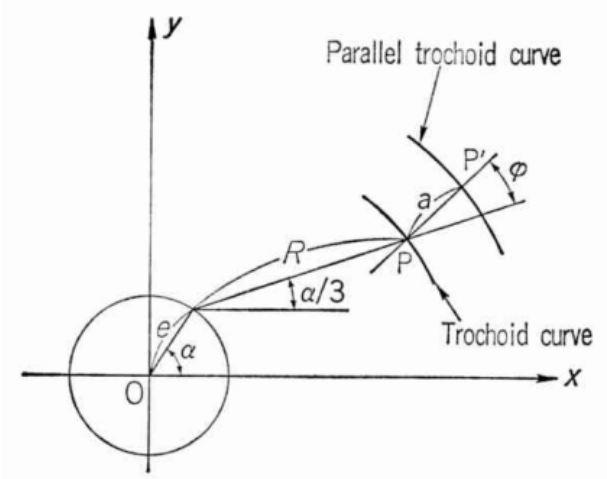


Figure 3.3: Geometrical generation of the stator curve [4].

The general equation of the peritrochoid is:

$$\begin{cases} x = e\cos(\alpha) + R\cos(\frac{\alpha}{m}) \\ y = e\sin(\alpha) + R\sin(\frac{\alpha}{m}) \end{cases} \tag{3.1}$$

Where α are the angles of rotation of the eccentric shaft and m is a parameter that controls the number of lobes of the trochoid. In this case it is set to 3 to have the two lobes configuration, so the equation (3.1) becomes:

$$\begin{cases} x = e\cos(\alpha) + R\cos(\frac{\alpha}{3}) \\ y = e\sin(\alpha) + R\sin(\frac{\alpha}{3}) \end{cases} \tag{3.2}$$

R and e are parameters related to the circles used to generate the curve, but in the actual engine geometry they coincide with the rotor radius and the eccentricity of its center. The ratio between them is called $K = R/e$ and it is fundamental in determining the shape of the peritrochoid. High values of K leads to a more circular shape (Figure 3.1c), while lower values enhances the lobes shape (Figure 3.1a). Usually it is set around $6 \sim 8$ for Wankel engines.
As already stated, the angle α is the generating angle (positive if rotating counterclockwise); to have the full peritrochoid shape it must vary from 0 to 6π degree. When designing the real engine, another parameter, called parallel transfer (a), must be taken into consideration. This is a small outward translation of the curve in order to accommodate the circular section of the apex seal and generate the so called "parallel trochoid curve" (Figure 3.3). This translation depends also by another parameter, called angle of oscillation (ϕ), according to:

$$\begin{cases} x = e\cos(\alpha) + R\cos(\frac{\alpha}{3}) + a\cos(\frac{\alpha}{3} + \phi) \\ y = e\sin(\alpha) + R\sin(\frac{\alpha}{3}) + a\sin(\frac{\alpha}{3} + \phi) \end{cases} \tag{3.3}$$

However, for practicality, this formula can be simplified with a small error to:

$$\begin{cases} x = e\cos(\alpha) + (R + a)\cos(\frac{\alpha}{3}) \\ y = e\sin(\alpha) + (R + a)\sin(\frac{\alpha}{3}) \end{cases} \tag{3.4}$$

The difference between equations (3.1), (3.2) and (3.4) is shown in Figure 3.4, which represents the shape for $\alpha = 0 \sim 2\pi$, meaning $\frac{1}{3}$ of total figure. It can be seen that the shapes are very similar and the black and the red line coincides in the regions where the

tip is perpendicular to the surface, while the red one diverge in the other regions.

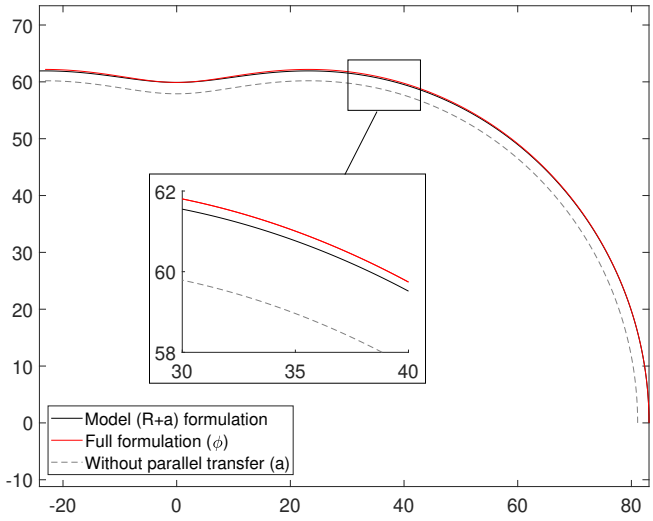


Figure 3.4: Stator equations comparison with MATLAB.

An important remark has to be pointed out regarding the implementation of this equation in the C++ code of the utility that will be covered in the subsequent section 3.2. As a matter of fact, for the model, it was chosen to rewrite the (3.4) in the following way:

$$\begin{cases} x = e\cos(\alpha) + (R' + S_p)\cos(\frac{\alpha}{3}) \\ y = e\sin(\alpha) + (R' + S_p)\sin(\frac{\alpha}{3}) \end{cases} \tag{3.5}$$

Where $R' = R + a'$.

The (3.4) and (3.5) are totally equivalent, but since a is not always known, and R' can be easily derived from the real rotor geometry (by measuring the distance from the rotor center to its tip), this formulation can be more practical in many situations. However, there is also another reason for this choice and it will be explained in the next section.

3.1.2. Rotor shape

The rotor shape is given by the contour of the inner envelope of the trochoid with the following expressions:

$$\begin{cases} X = R\cos(2v) + \dfrac{3e^2}{2R}(\cos 8v - \cos 4v) + e\sqrt{(1 - \dfrac{9e^2}{R^2}\sin^2 3v)}(\cos 5v + \cos v) \\ Y = R\sin(2v) + \dfrac{3e^2}{2R}(\sin 8v + \sin 4v) + e\sqrt{(1 - \dfrac{9e^2}{R^2}\sin^2 3v)}(\sin 5v - \sin v) \end{cases} \tag{3.6}$$

Where X and Y are cyclic functions of period 2π and, to obtain only the internal contour (corresponding to the rotor shape), the following intervals of v must be considered:

$$\begin{cases} v = \dfrac{1}{6}\pi \sim \dfrac{1}{2}\pi \\ v = \dfrac{5}{6}\pi \sim \dfrac{7}{6}\pi \\ v = \dfrac{3}{2}\pi \sim \dfrac{11}{6}\pi \end{cases} \tag{3.7}$$

Also in this case for the real engine geometry a parallel transfer a' should be considered. The parameter a' is defined as $a' = a - S_p$ where S_p is the minimum clearance between the rotor and the rotor housing, chosen considering bearing clearance, thermal deformation, and errors in manufacturing; usually its value is set to 0.5 mm. With this consideration, the equations (3.6) becomes:

$$\begin{cases} X = R\cos(2v) + \dfrac{3e^2}{2R}(\cos 8v - \cos 4v) \\ \qquad + e\sqrt{\left(1 - \dfrac{9e^2}{R^2}\sin^2 3v\right)}(\cos 5v + \cos v) \\ \qquad + \dfrac{3}{2}\dfrac{ea'}{R}(\cos 5v - \cos v) + a'\cos 2v\sqrt{\left(1 - \dfrac{9e^2}{R^2}\sin^2 3v\right)} \\ Y = R\sin(2v) + \dfrac{3e^2}{2R}(\sin 8v + \sin 4v) \\ \qquad + e\sqrt{\left(1 - \dfrac{9e^2}{R^2}\sin^2 3v\right)}(\sin 5v - \sin v) \\ \qquad + \dfrac{3}{2}\dfrac{ea'}{R}(\sin 5v + \sin v) + a'\sin 2v\sqrt{\left(1 - \dfrac{9e^2}{R^2}\sin^2 3v\right)} \end{cases} \tag{3.8}$$

This is the correct way to represent the rotor shape because it includes also a certain degree

of "opening" between the rotor flanks (red curve in Figure 3.5), in order to simulate the width of the apex seals. As it was done in the previous section, also in this case a remark is made with regard to the adaptation made on the equation (3.8) in the C++ code in Openfoam®. In fact, for the utility, the equation that was employed is:

$$\begin{cases} X = R'\cos(2v) + \dfrac{3e^2}{2R'}(\cos 8v - \cos 4v) + e\sqrt{(1 - \dfrac{9e^2}{R'^2}\sin^2 3v)}(\cos 5v + \cos v) \\ Y = R'\sin(2v) + \dfrac{3e^2}{2R'}(\sin 8v + \sin 4v) + e\sqrt{(1 - \dfrac{9e^2}{R'^2}\sin^2 3v)}(\sin 5v - \sin v) \end{cases} \tag{3.9}$$

Where $R' = R + a'$.
In addition to the observation made in 3.1.1, the choice to use R' in this code is done to simulate the apex seals considering them as zero-thickness patches, to allow the further possibility to implement a blow-by model exploiting their porosity. As a matter of fact, using the (3.9) the shape of the rotor is completely closed and the apex seals are not modeled. Unlike what was done for the stator, this decision can lead to a difference with respect to the real engine geometry; however, this error turns out to be very small. Figure 3.5 shows the comparison between the shapes obtained from (3.6), (3.8) and (3.9).

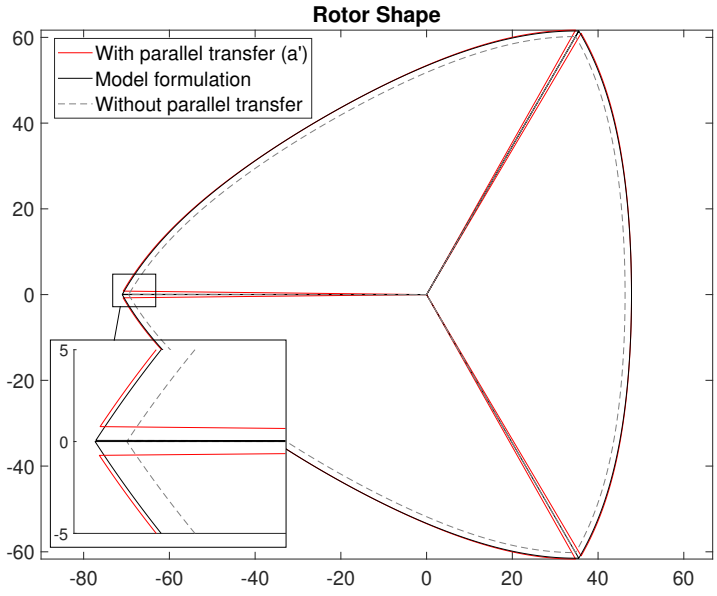


Figure 3.5: Rotor equations comparison with MATLAB.

3.1.3. Oscillation angle

The oscillation angle ϕ is the angle between the apex seals mean line direction and the normal to the surface of the rotor housing (see Figure 3.1), and it greatly affects the performance of the engine. It can be calculated by the following equation:

$$\phi = \frac{3e\cos\frac{2}{3}\alpha + R}{\sqrt{9e^2 + R^2 + 6eR\cos\frac{2}{3}\alpha}} \tag{3.10}$$

Since $\phi = \phi(\alpha, K)$ (see Figure 3.6) by deriving and applying the condition $\frac{d\phi}{d\alpha} = 0$ we can find its maximum:

$$\phi_{\max} = \frac{3}{K} \tag{3.11}$$

This means that for a fixed engine geometry (since K depends only on R and e) the maximum oscillation angle is determined. This parameter will be later used in the calculation of the chambers volume and the CR.

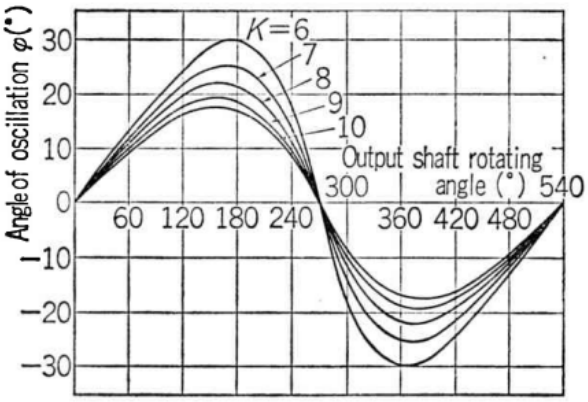


Figure 3.6: Oscillation Angle curve with respect of α and effect of K.

3.1.4. Rotor motion

All the points of the rotor shape, generated by (3.9), must perform a rototranslation to replicate the motion of the rotor inside the chamber. In particular, the shape must rotate around the rotor center, which is in turn in rotating three times faster around the origin of the coordinate system, following a circumference of radius e. Assuming a constant angular velocity, the equation that describe this motion is the following:

$$\begin{cases} X_{translated} = -e\cos\theta + X\cos\left(\dfrac{\theta}{3}\right) + Y\sin\left(\dfrac{\theta}{3}\right) \\ Y_{translated} = e\sin\theta + X\sin\left(\dfrac{\theta}{3}\right) + Y\cos\left(\dfrac{\theta}{3}\right) \end{cases} \tag{3.12}$$

where θ is the desired ESA and X and Y are the coordinates of the points of the rotor shape from (3.9). Note that since the arguments of the sin and cos formulas are in a 3:1 ratio, three full rotation of the rotor center around the origin are needed in order to complete one complete rotation of the points of the rotor. Since it was chosen to set a clockwise rotation of the rotor, also θ is defined positive when it is clockwise, contrarily to α, the generating angle of the trochoid shape, despite them representing basically the same physical quantity, i.e. the ESA.

3.1.5. Engine Displacement Law

The engine displacement law is the volume obtained by multiplying the chamber area enclosed between the rotor and the stator curve at each position, multiplied by the thickness of the rotor. The volume trend, considering a and a', is represented with a certain accuracy by the following equation [4]:

$$\begin{aligned} V &= A \cdot b = \\ &= V_{min} + \frac{3\sqrt{3}}{2} e\,(2R_1 + R_2)\,b\left(1 - \sin\frac{2}{3}\alpha + \frac{\pi}{6}\right) \end{aligned} \tag{3.13}$$

where

$$V_{min} = \left\{ \frac{\pi}{3}e^2 + \frac{(R_1^2 - R_2^2)}{3}\pi + 2eR_2\cos\phi_{max} + \left(\frac{2}{9}R_2^2 + 4e^2\right)\phi_{max} - \frac{\sqrt{3}}{2}e\left(R_1^2 - R_2^2\right) \right\} b \tag{3.14}$$

and

$$\begin{cases} R_1 = R + a \\ R_2 = R + a' \end{cases} \tag{3.15}$$

3.1.6. Compression Ratio

The CR of the rotary engine, due to the design of its chambers, is usually relatively low. The maximum value that it can assume coincides with the theoretical compression ratio

suggested by [4]:

$$\epsilon_{th} = \frac{2eR\cos\phi_{max} + \left(\frac{2}{9}R^2 + 4e^2\right)\phi_{max} + \frac{\pi}{3}e^2 + \frac{3\sqrt{3}}{2}eR}{2eR\cos\phi_{max} + \left(\frac{2}{9}R^2 + 4e^2\right)\phi_{max} + \frac{\pi}{3}e^2 - \frac{3\sqrt{3}}{2}eR} \tag{3.16}$$

As it can be seen the theoretical CR depends on K and ϕ_{max} only (see figure 3.7). However, it was found out that this formulation lead to unrealistic and very large values of the CR, so, to have a better estimation of the compression ratio (considering the parallel transfer), especially if we will have to compare it with the mesh one coming from OpenFOAM, it can be calculated as:

$$\begin{aligned} \epsilon_{geom} &= \frac{V_{max}}{V_{min}} = \\ &= \frac{V_{min} + V_{stroke}}{V_{min}} = \\ &= \frac{V_{min} + \sqrt{3}e\,(2R_1 + R_2)\,b}{V_{min}} \end{aligned} \tag{3.17}$$

Where V_{min} and R_1 and R_2 are again obtained from (3.14) and (3.15).
It is important to notice that these CR does not take into account the volumes of the various recesses of the engine, which should be added both at the numerator and at the denominator, to have a more accurate estimation.

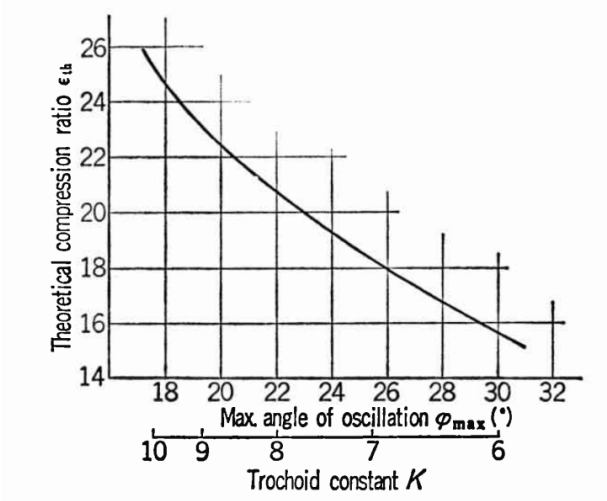
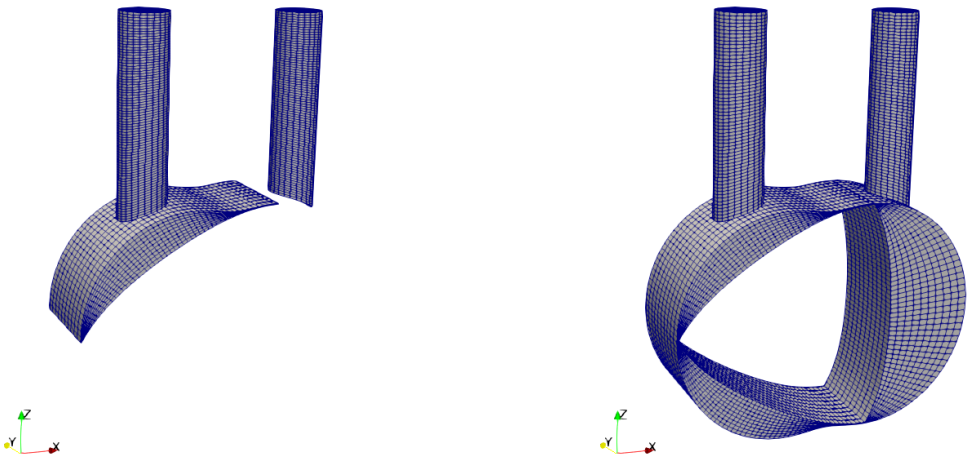


Figure 3.7: Compression ratio with respect to oscillation angle and trochoid constant from [4].

3.2. Utilities

The aim of this chapter is to outline the grid generation methodology and the utilities used to develop the model of a peripherally ported Wankel engine in OpenFOAM 8. It was chosen to develop two different utilities written in C++ code, one for simulating just a single chamber rotating (figure 3.8a), and one for the full Wankel geometry, meaning three rotating chambers (figure 3.8b). For the first case the utility is named `createWankelEngineMesh`, that is recalling another utility named `makeWankelMesh`, while for the second case the utilities are `createWankelEngineFullMesh` and `makeWankelFullMesh`. For both the configurations the coupling with the ducts meshes, created with the `makeWankelIntakeExhaust` utility, is done through the Arbitrary Coupled Mesh Interface (ACMI). In the next subsections the working principle of these tools is presented, highlighting how the equations previously presented are utilized to generate the grid.



(a) (b)

Figure 3.8: To the left the mesh of the single chamber configuration, to the right the one for the full configuration.

3.2.1. `createWankelEngineMesh`

This utility is located in the `\$POLIMI\_UTILS/preProcessing` folder. The approach used by the `createWankelEngineMesh` to generate the mesh is to create the grid at

many discrete timesteps over the total angular extension of the simulation domain and then, at each one of these locations, construct a local mesh, using the `makeWankelMesh` and the `makeWankeIntakeExhaust`. The amount of meshes created are defined by a parameter called `thetaResolution`. The mesh is then linearly interpolated between these discrete positions by using the interpolation scheme `dynamicInterpolatedFvMesh`, which is specified in the `constant/dynamicMeshDict` dictionary. During the simulation, this algorithm computes the mesh's state at intermediate time steps by interpolating the mesh node positions between the pre-generated meshes. Since the interpolation process is generally less computationally expensive than solving the transport equations for every crank angles, this aproach can be more efficient. Also, it provides greater control over the mesh quality at specific time instances, as the pre-generated meshes can be optimized individually. However, this interpolation lead to a possible error if the chosen resolution parameter is too large (see 3.1.5). It is important to remember that the possibility of pre-allocating various meshes at each timestep is made possible by the presence of a prescribed motion of the engine, meaning that the rotor points trajectories are known a priori.
In addition, the `createWankelEngineMesh` utility also implements the possibility to use the `snappyHexMesh` tool to import the recess and the real ducts geometry from an `.stl` file, since the two basic utilities previously mentioned only create the rotor contour and cylindrical ducts. The `createWankelEngineMesh` implements the coupling with the ducts mesh with its feature `mergeMesh`, which triggers some additional line of codes that, after running the commands to create the set of meshes (ducts and chamber), generate the proper interface in order for the ACMI coupling to work with the commands `topoSet` and `createBaffles`.

3.2.2. makeWankelMesh

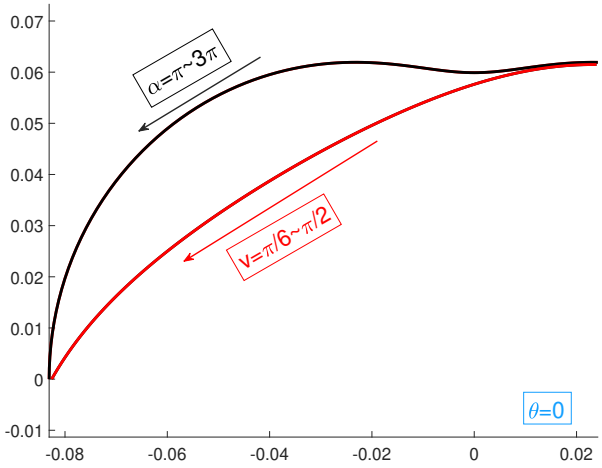
This utility is located in `$POLIMI_UTILS/mesh`. Its goal is to create the local mesh for every discrete position recalled by the `createWankelEngineMesh` function. Briefly, this tool firstly creates the desired rotor shape (a single edge, since this is the single chamber case), then it rotates it to the desired angular position and it builds the stator shape around it. The `makeWankelMesh` folder, other than the implementation `.C` file of the utilities itself, contains some headers files to carry out this implementation:

- `readGeometryAndMeshData.H` Responsible of reading and importing the dictionary containing the geometrical dimensions of the engine and the toggles for some additional features of the `createWankelEngineMesh`, specifically from the files *meshParameters* and *wankelGeometry*, located in the case directory.

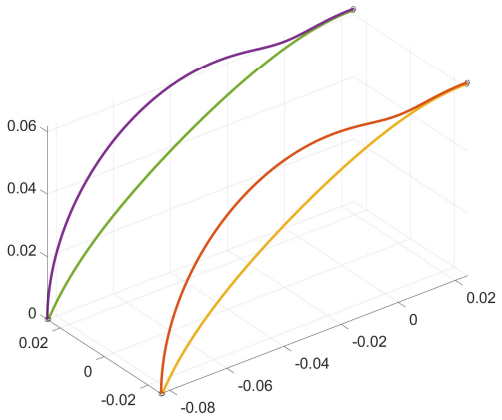
- `createRotorPoints.H` Creates the curves of rotor geometry and rotates its shape to the desired angular where the mesh has to be created. To better clarify the concept, given the single chamber configuration, the code creates only one of the three rotor flanks, substituting the interval $v = \frac{\pi}{6} \sim \frac{\pi}{2}$ in equation (3.9) and then, using the equation (3.12), performs a rototranslation of all the points of this curve to the desired θ angle. Since the rotor is rotating in a clockwise direction the θ angle is set to start from the left part of the x-axis and it's positive clockwise too.

- `createStatorPoints.H` Creates lists of points and curves of the stator geometry using equation (3.5). To elaborate further, the code constructs the stator curve over the rotor one by properly selecting the intervals in which to define the equation. For example, for the position $\theta = 0$, the stator curve overlaps perfectly the rotor flank if it is defined for an interval that goes from π to 3π. Then, for the following position, as mentioned previously, the rotor curve is rotated by a certain θ', while the stator shape is generated on top of it by shifting in a clockwise manner (of the desired θ') the definition interval, that will become $\pi - \theta' \sim 3\pi - \theta'$. The fact that the interval extent is $\pi \sim 3\pi$ can seem counter intuitive, but this happens because the equation (3.5) is defined through the α angle, that, in contrast of θ, rotates in the counterclockwise direction to define the peritrochoidal shape. To further elaborate, if θ is defined in the top-left quadrant and is rotating clockwise, that zone is also the region where the stator curve is generated by the interval π to 3π of α, which starts from the top-right quadrant and rotates counterclockwise (both the angles to complete the full rotation need 1080 ESA). Also, intuitively, a single rotor flank will cover $\frac{1}{3}$ of the total stator extension (defined by an interval of 6π), that is exactly 2π. Figure 3.9b shows an example of the polylines created by `createRotorPoints.H` and `createStatorPoints.H`.

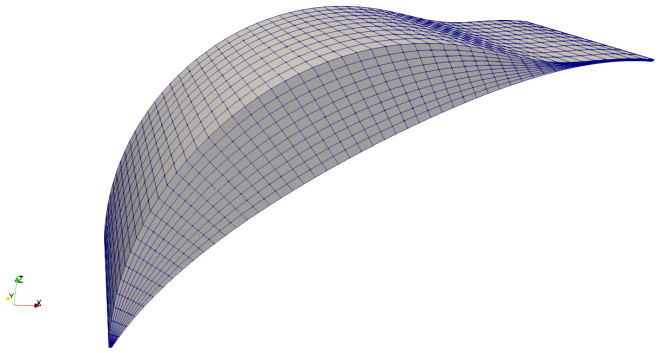
- `writeMeshFiles.H` It takes the outputs from `createRotorPoints.H` and `createStatorPoints.H` and construct, in the `constant` folder of the case, three files named *wankelPoints*, *wankelBlocks* and *wankelPolyLines*. These files are the entries for the dictionary `blockMeshDict` for the OpenFOAM® utility `blockMesh`, which is the tool used to create a structured hexahedral mesh. The `cellZone` created in this way is named `rotorCells`. The final rotor mesh is shown in Figure 3.9c.



(a) Polylines at $\theta = 0$.



(b) 3-D view of polylines at $\theta = 0$.

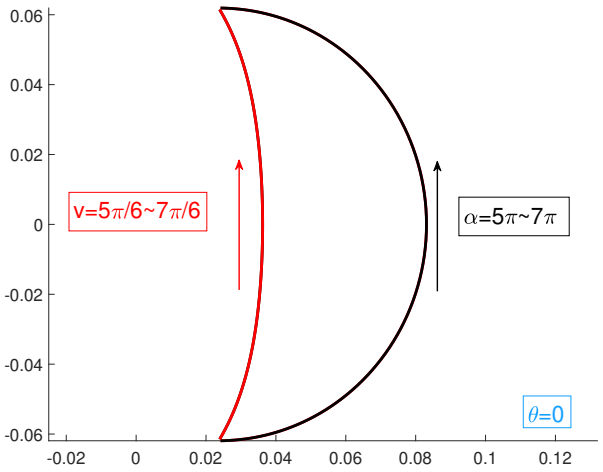


(c) Mesh at $\theta = 0$

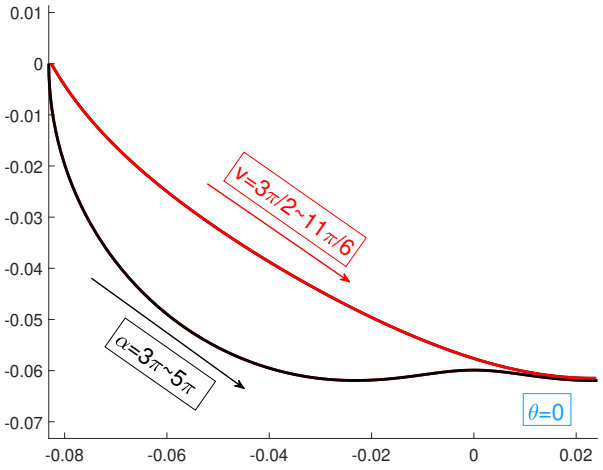
Figure 3.9: In this picture it is possible to see the polynomial curves and the relative mesh for a single chamber for $\theta = 0$. In the first image, the generating angles intervals are highlighted.

3.2.3. `makeWankelFullMesh` and `createWankelEngineFullMesh`

`makeWankelFullMesh` and `createWankelEngineFullMesh`, as already stated, are related to the three chambers configuration, but their coding is pretty similar to the one chamber configuration. The utility `makeWankelFullMesh` works exactly like the `makeWankelMesh`, with the only difference that it contains one `createRotorPoints.H` and one `createStatorPoints.H` for each one of the three chambers, creating a total of nine entries files for the `blockMeshDict`. The intervals used to generate the additional two chambers are shown in Figure 3.10, while Figure 3.11 presents the 3D plot of the polylines and the nomenclature of the cellZones created.

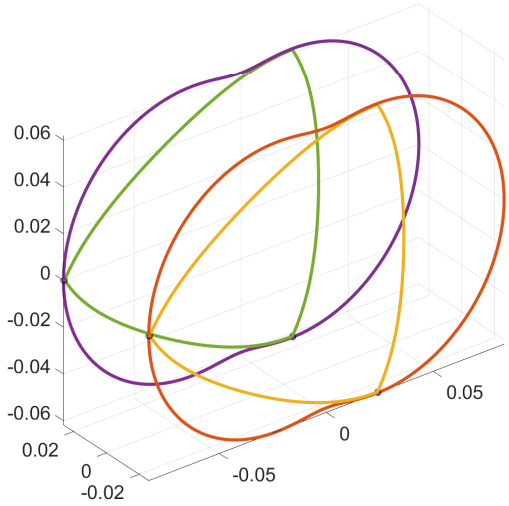


(a) Polylines of the second chamber at $\theta = 0$.

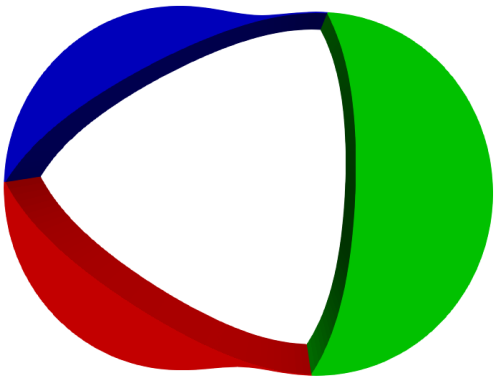


(b) Polylines of the third chamber at $\theta = 0$.

Figure 3.10



(a) 3D polylines plot for the 3-Chambers configuration at $theta = 0$.



(b) In blue: the first chamber cellZone, namely rotorCells1. In green rotorCells2 and in red rotorCells3.

Figure 3.11: These pictures show the polynomial lines and generation angles intervals used to build the other two chambers.

3.2.4. `createWankelIntakeExhaust`

`createWankelIntakeExhaust` is the utility used to create the intake and exhaust ducts, specifically, the cell zones named `intakeCells` and `exhaustCells`. They are created as cylindrical and their location is set by the angular position θ. For example, to set the position of the intake ducts, the ESA at which the intake port starts opening must be

specified in the *meshParameters* dictionary (see 3.3.2), together with the other geometrical and resolution parameters, such as diameter, length, number of cells and so on. The `createWankelIntakeExhaust` creates also the spark plug recesses, since they were modeled and positioned along the housing perimeter in a similar way with respect to the pipes, and they are treated similarly also by the ACMI coupling[2].

3.3. Dictionaries

This section will cover the dictionaries setup, in order to properly define what are the entries that the user must chose for the creation of the mesh.

3.3.1. *wankelGeometry*

wankelGeometry file can be placed outside the case directory, given that the proper path to it is defined in the *engineGeometry*, and it contains the geometric dimensions of the engine. Specifically, the entries are:

- `thickness` The rotor thickness (b)
- `rotorRadius` The rotor radius, that in this case has to be set with the catalogue value R, plus a' (see 3.1.2).
- `eccentricity` The eccentricity (e)
- `clearance` The minimum clearance (S_p)

3.3.2. *meshParameters*

This dictionary contains the parameters needed to generate the grid and assign to it a certain resolution. In particular, the first entries are related to the rotor grid and are:

- `y0stator` This parameter is read by the `makeWankelMesh` and used to place the grid in the global reference system. Since y direction is the one along the thickness of the chamber, if `y0stator` is set to 0 the mesh will be placed with half thickness in the negative y-quadrant and half in the positive y-quadrant.
- N_x, N_y, N_z They define the number of cells along the 3-D coordinate system as shown in Figure 3.12. N_x is the number of cells along the radial direction, or the clearance direction. N_y is the number of cells along the thickness of the rotor and N_z is the

[2]Also the spark plug recess will need an ACMI interface to be coupled cyclically, at every rotation, with the chamber region

number of cells along the circumferential extensions of the chamber.

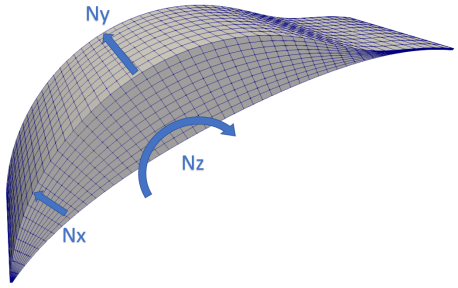


Figure 3.12: Mesh resolution parameters.

- `tethaResolution` The parameter responsible of the number of discrete meshes created by the `createWankelEngineFullMesh`, e.g. if it is set to 1 the meshes are created at every ESA and the grid between two consecutive position is interpolated (see 3.2.1). As will be shown later, this is an important parameter that can influence the accuracy of the results.
- `pLineResolution` This parameter defines the resolution of the polylines, meaning the number of points composing each polyline shown in Figure 3.9b.
- `startTime` **and** `endTime` These specify the operating interval, in ESA, in which the mesh must be created.
- `includeRotorGeometry` Can be set to `on` and `off` depending on whether or not the recess shape of the rotor should be considered. If set to `on`, the path to the `.stl` file must be indicated. Then, the "morphing" of the recess on the rotor surface is done with the help of the `snappyHexMesh` tool.

The following entries, instead, are related to the ducts mesh creation. Both the intake and the exhaust are created as cylindical and will have a 5 blocks mesh structure (Figure 3.13).

- `thetaIntakeDeg` **and** `thetaExhaustDeg` These angles are used to locate the ducts along the stator surface. It is important to notice that the angles are taken counter-clockwise in a X-Z coordinate system centred in the origin, where the stator center is placed too. The angles must locate the position where the ducts closes by the perspective of the rotor which is rotating clockwise.
- `dIntake` **and** `dExhaust` Respectively, the diameters of intake and exhaust.
- `lIntake` **and** `lExhaust` The length of respectively Intake and exhaust.
- $N_{xy}Intake$**,** $N_{xy}Exhaust$ The resolution of the grid in the x-y plane.

- $N_zIntake$, $N_zExhaust$ The resolution in the Z direction.
- `intakeOffset` **and** `exhaustOffset` These two entries allow to move the axis of the ducts along the y-direction in an asymmetrical position with respect to the rotor chamber.
- `sqRatioIntake` **and** `sqRatioExhaust` These parameters determines how much percentage of the ducts radius is occupied by the square block.
- `mergeMesh` It can be set to `true` or `false` depending on whether or not the ducts mesh has to be included. If set to `true`, the path to the ducts case has to specified.
- `snapHead` If this parameter is set to `on` and the `.stl` of the engine head is given, the real ducts geometry can be reproduced.

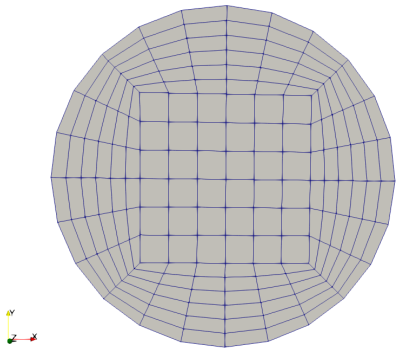


Figure 3.13: This picture shows the mesh composed of five blocks, with a square block in the center and four other blocks surrounding it.

In addition, for the spark plugs we have:

- `nSparks` The number of spark plugs.
- `dSparks` Diameter of spark plugs recess.
- `hSparks` Depth number of spark plugs recess.
- `ySparks` Locate the position of spark plugs on the x-axis, similarly to `y0stator`.
- `thetaSparksDeg` Used to locate the spark plugs on the housing. The same consideration done for `thetaIntakeDeg` and `thetaExhaustDeg` are valid.
- `sqRatioSparks` Since the spark plug recess mesh is modeled as the ducts one, this parameter has the same function of `sqRatioIntake` and `sqRatioExhaust`.

- $n_{xy}Sparks, n_zSparks$ Parameters for the mesh resolution.

3.4. Additional utilities and dictionaries

The utilities presented in the previous chapter were tailor-made for Wankel engines. Here below, instead, the other crucial tools employed for the mesh generation that were already part of OpenFOAM libraries are clarified.

3.4.1. `blockMesh` and `blockMeshDict`

`blockMesh` is the final implementation of the mesh creation procedure presented in the previous sections. Despite its rather simple execution it is capable to create many different shapes in a flexible manner, allowing to create also geometries composed by curved edges. The principle behind `blockMesh` is to decompose the domain geometry into a set of 1 or more three dimensional, hexahedral blocks. Edges of the blocks can be straight lines, arcs or splines. The mesh is specified as a number of cells in each direction of the block, which are sufficient informations for blockMesh to generate the mesh data [32]. Each block of the geometry is defined by 8 vertices, one at each corner of a hexahedron. The vertices are written in a list so that each vertex can be accessed using its label, remembering that OpenFOAM always uses the C++ convention that the first element of the list has label '0'.

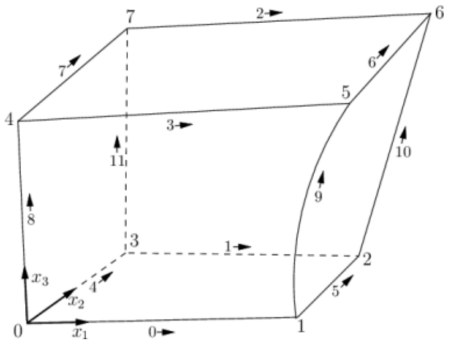


Figure 3.14: `blockMesh` block.

Figure 3.14 shows the single block structure, where the edge 1-5 is curved to highlight the flexibility of this tool regarding edge shapes. Each block has a right-handed local coordinate system, that is defined by the order with which the vertexes are listed in the block definition, specifically:

- Vertex 0 is the axis origin.

- Vertices 0-1 describe the x_1 direction.
- Vertices 1-2 describe the x_2 direction.
- Vertices 0, 1, 2, 3 define the plane $x_3 = 0$.
- Vertices 0-4 descibe the x_3 direction. Point 4 is obtained moving along this direction starting from 0.
- Vertices 5, 6, and 7 are similarly found by moving in the x_3 direction from vertices 1, 2, and 3 respectively.

The `blockMesh` read the inputs from the `system/blockMeshDict` dictionary, that is arranged as follows:

```
convertToMeters 1; \\scaling factor
vertices
(
   #include "$FOAM_CASE/constant/wankelpoints1"
   #include "$FOAM_CASE/constant/wankelpoints2"
   #include "$FOAM_CASE/constant/wankelpoints3"
);

blocks
(
    #include "$FOAM_CASE/constant/wankelBlocks1"
    #include "$FOAM_CASE/constant/wankelBlocks2"
    #include "$FOAM_CASE/constant/wankelBlocks3"
);

edges
(
#include "$FOAM_CASE/constant/wankelPolyLines1"
#include "$FOAM_CASE/constant/wankelPolyLines2"
#include "$FOAM_CASE/constant/wankelPolyLines3"
);

boundary
(
    rotor
    {
        type wall;
```

```
        faces
        (
            (0 4 7 3)
        );
    }
    seal1
    {
        type wall;
        faces
        (
            (2 6 5 1)
        );
    }
    //other patches here...//

);

mergePatchPairs
(
);
```

The points are usually given as an ordered list directly in the `blockMeshDict`, but in this case the points are included with the file *wankelPoints*. The `blocks` sub dictionary, takes the *wankelBlocks* entry, that for each chamber contains:

```
hex (0 1 2 3 4 5 6 7) rotorCells1 (10 10 50) simpleGrading (1 1 1)
```

Where `hex` stands for hexaedral cells and the following brackets contains the vertices names, ordered respecting the rules explained previously. Additionally, the word `rotorCells1` represents the name of the cellZone of the specific chamber (1,2 or 3), which precedes the definition of N_x, N_y and N_z (defined in the `meshParameters` dictionary). The final entry, instead, gives the cell expansion ratios for each direction in the block, which enables the mesh to be graded, or refined, in specified directions, but this tool was left to the standard value (no grading). The `edges` dictionary takes different possible entries, like arc, simpleSpline, polyLine, polySpline. In this work the polyLine entry was used, specifying the file wankelPolyLines, expressed as:

```
polyLine 0 4
(
    (x1 y1 z1)
```

```
    (x2 y2 z2)
    ...
)
polyLine 1 5
(
    ...
)
    ...
```

The first element is the type of curve used, while the following numbers are the names of the vertices between which the interpolated points of the following list are included. Each *wankelPolylines* will contains a total of four polylines, since this is the number needed to create one chamber shape (see Figure 3.11a. The `boundary` subdictionary contains the boundary of the mesh, breaking the mesh into patches and associating a name to them. As it can be seen each patch has a `type` and a `faces` entries. The first one gives different options, but in the specific case of this thesis work the `wall` type was applied, because in case of turbulence modeling, especially if wall functions were used, this type is particularly useful, since distance from the wall of the cell centres next to the wall are stored as part of the patch. The other patches are not shown here but they will be shown later, when the setup of the case will be presented. The `convertToMeters` is just a scaling factor to refer the dimension to meters $[m]$, and the `mergePatchPairs`, that is used to merge different blocks together, was not utilized, since the chambers must remain distinct one from each other.

3.4.2. Utilities and dictionaries related to ACMI coupling

As already specified, the meshes from the two different cases of the ducts and the chambers are coupled using the ACMI coupling. This coupling method, an evolution of Arbitrary Mesh Interface (AMI), is designed to handle the interaction between non-conformal, non-matching, and rotating mesh regions. The ACMI approach is particularly useful in simulations involving complex geometries, such as Wankel engines, where the chamber and ducts meshes may not have a one-to-one correspondence and can undergo relative motion or rotation. The ACMI coupling works by creating a patch-based interface between the two mesh regions. This interface allows for the exchange of information, such as flow variables and boundary conditions, between the two regions without requiring to have matching cell faces. In OpenFOAM, ACMI coupling is implemented using the cyclicACMI boundary condition (see 4.3), which is specifically designed form meshes that interacts and exchange fluids cyclically. The methodology to properly set up the ACMI involves

the creations of some interfaces trough the utilities `topoSet` and `createBaffles`, which are run after the mesh generation by the `createWankelEngineMesh` utility. These utilities work by reading their relative dictionaries, `topoSetDict` and `createBafflesDict`, which are included in the `system` of the case.
The purpose of the first file is to create face zones that will be used later by the createBaffles utility to define the ACMI interfaces. It does this by creating face sets from existing patches (i.e. stator, intakeInterface, exhaustInterface, and sparkInterface0[3]) and then converting these face sets into face zones (statorFaces and portsFaces), which are groups of faces with an associated orientation, which are used to define a region in the mesh that will be treated specially by the solver.
The purpose of `createBafflesDict`, instead, is to create the ACMI interfaces (and the associated blockage patches) in the mesh using the face zones created by topoSetDict. The ACMI interfaces are created for the stator and ports (intake, exhaust, and spark plug interfaces). The createBaffles utility reads this file and modifies the mesh accordingly. For each ACMI interface (ACMI1 and ACMI2, respectively related to the stator surface and the ducts ports), there are two patches created: a "couple" patch and a "blockage" patch. The "couple" patches (`ACMI1_couple` and `ACMI2_couple`) are the actual ACMI interfaces, and they are responsible for interpolating the flow variables between the non-conformal meshes. The "blockage" patches (`ACMI1_blockage` and `ACMI2_blockage`) are created to represent the region where there is no overlap between the meshes, and they are treated as walls. Since they are walls, no-slip boundary conditions will be applied to the velocity field, and other fields will have their respective wall boundary conditions applied. The `createBafflesDict`, together with the setup of the interfaces mentioned above, has an entry named `internalFacesOnly`, which must be set to false, since we are dealing with the boundary faces of the stator.
This last point is particularly critical in the simulation setup of the Wankel engine and lead the first simulations results to some inconsistencies with respect of the conservation laws. In fact, mass reduction was observed after the IPC (when theoretically, the mass should have remained constant, as a blowby model was not set up) and there was non-conservation of chemical species. The reason for this behavior can be found recalling the equation (2.61), which highlight that an additional convective term appears in case of dynamic meshes. This is due to the fact that the equation (2.61) is implemented also in OpenFOAM and the new flux correction component is called `meshPhi`. In case of Wankel engine this correction leads to an error related for the housing boundary cells. Figure 3.15 shows the deformation of a cell caused by the chamber that is compressing.

[3]this is related to the interface between the chamber and the spark plug recess.

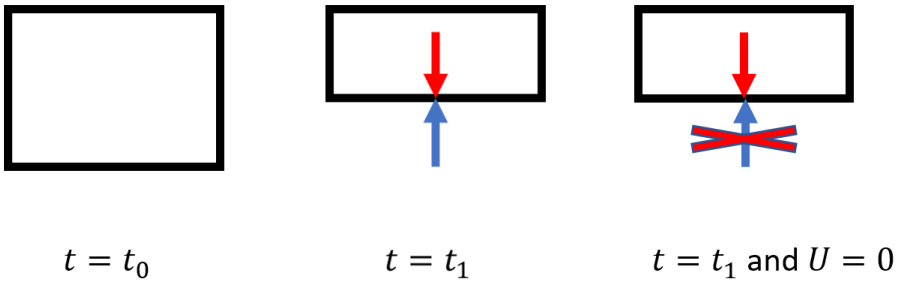


Figure 3.15: Deformation of stator boundary cells.

It can be seen that the velocity U of the boundary cell U (blue arrow pointing inward) is properly balanced by the velocity U_b of the mesh (red arrow pointing outward) after the deformation. However, the velocity of the boundary cells of the stator are set to be null ($U = 0$), since it is stationary and this will cause the term related to the contribution of U of the mass conservation (2.61) to become zero, while the correction term will remain present. The `meshPhi` will become the only convective term inside the equation, leading to an inconsistency of the conservation law and a consequent nonphysical reduction of mass. This issue was solved by setting inside the `constant/engineGeometry` file:

```
fixMeshFlux true;

patchesToFix
(
    ACMI1_blockage
    ACMI2_blockage
    ACMI1_couple
    ACMI2_couple
);
```

The purpose of `fixMeshFlux` is exactly to cancel this residual term related to `meshPhi` in correspondence with the specified ACMI patches. This helps to maintain the conservation of mass, momentum, and other quantities when transferring data between the non-conformal mesh regions, allowing to obtain a perfectly flat mass curve during the crank angles where the chamber is completely sealed.

3.5. Model consistency

3.5.1. AIE 225CS

Since this thesis work was intended as a study on UAV rotary engines, to verify the consistency of the model, meaning the engine displacement law and the compression ratio, the specific data of the AIE225CS engine was used. The AIE (UK) 225CS is a single-rotor peripheral port injected twin-spark engine that can operate on multiple fuels. When configured for aerospace use it has a nominal peak power output of 30kW with a core mass (excluding ancillaries) of 10kg [23]. Because of this and its compactness, appreciable from Figure 3.16, it's clear why this engine is perfectly suited for the production of ultra-lightweight aerial vehicles. Also, it can be observed from Figure 3.17, the flat torque curve characteristic and the range of operation, showing high power output at high rpm values. One of the key characteristic of this engine is the SPARCS system: a patented liquid cooling system, which features are comprehensively covered in [5] and [11]. However, gas blow-by and combustion are out off topic of the present work and can only be the subject of further studies. Table 3.2 reports the main characteristic of the engine.

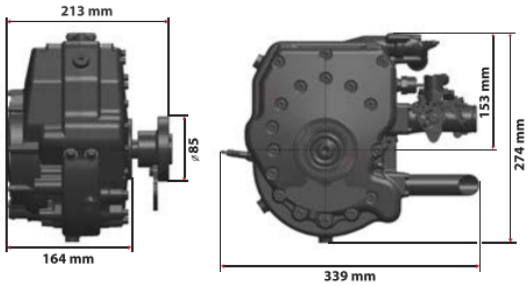


Figure 3.16: Dimensions of AIE225 CS.

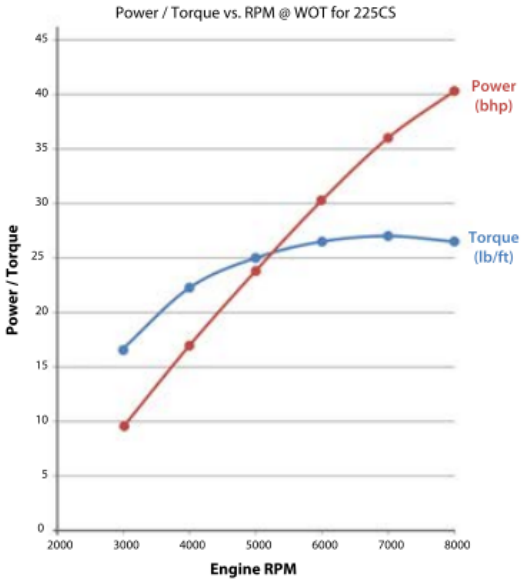


Figure 3.17: Power and torque curve of AIE225 CS.

AIE 225 CS specs			
Generating radius	R	69.5	mm
Eccentricity	e	11.6	mm
Parallel Transfer	a	2	mm
Width	b	51.941	mm
Clearance	S_p	0.5	mm
Number of Rotors	1		
Compression Ratio	9.6:1		
Weight	10		Kg
Power	30		KW
Torque	36.6Nm@8000rpm		
Fuel	AvGas / Gasoline / Heavy Fuels		
Cooling system	liquid cooled patented SPARCS		
Ignition	Twin spark plug		
Oil system	Digitally optimized lubrication		
Port timing			
Port	Opens	Closes	Units
Intake	71 before TDC	60 After BDC	ESA
Exhaust	69 before TDC	57 After BDC	ESA
Effective Port Overlap	128		ESA

Table 3.2: Main AIE225 CS specs.

3.5.2. Engine displacement law and CR

The consistency of the model for what concerns for the engine displacement law is made by comparing the results from the approximated analytical expression (3.13) from Yamamoto [4] with the one of the `vCyl.theta` file obtained from the OpenFOAM® mesh.

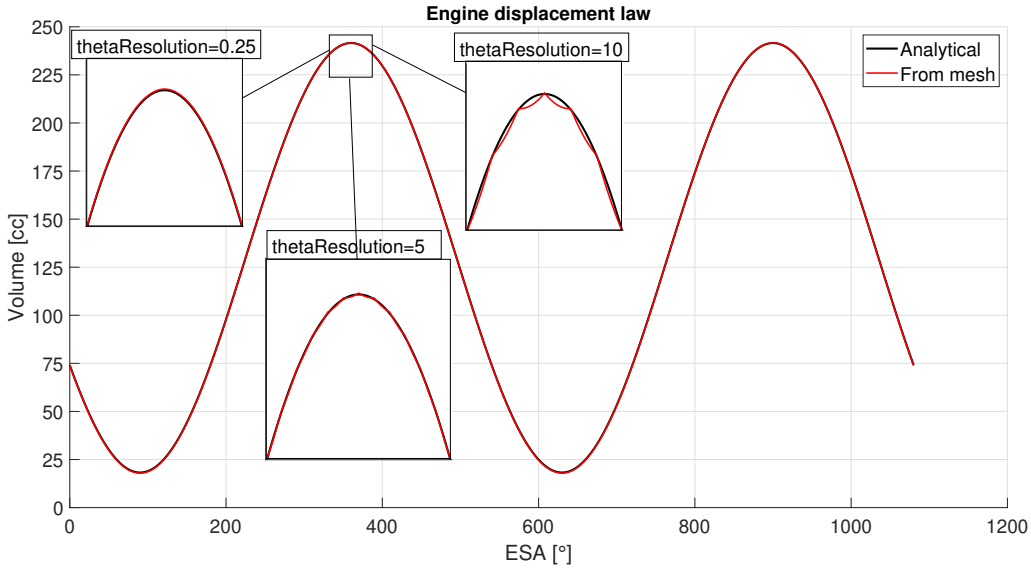
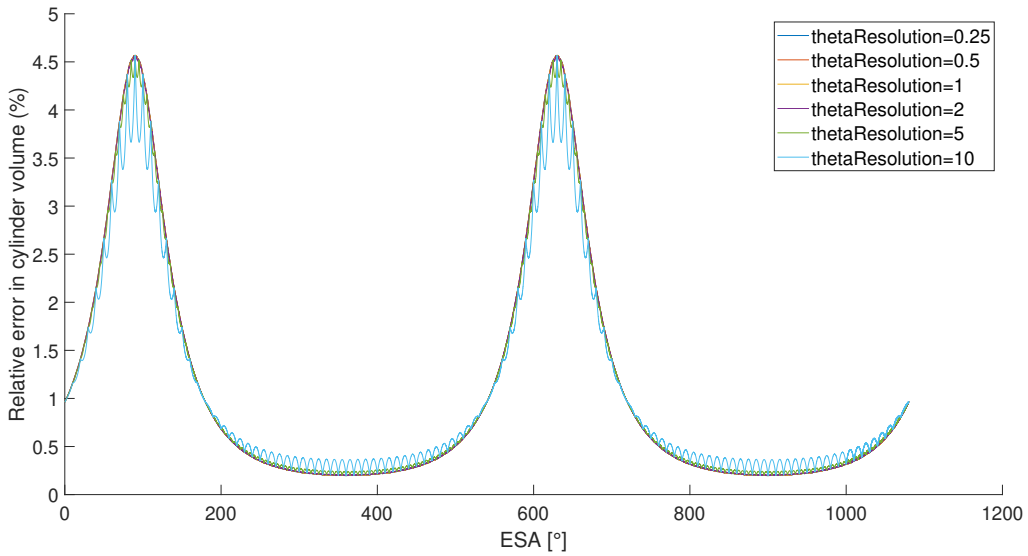


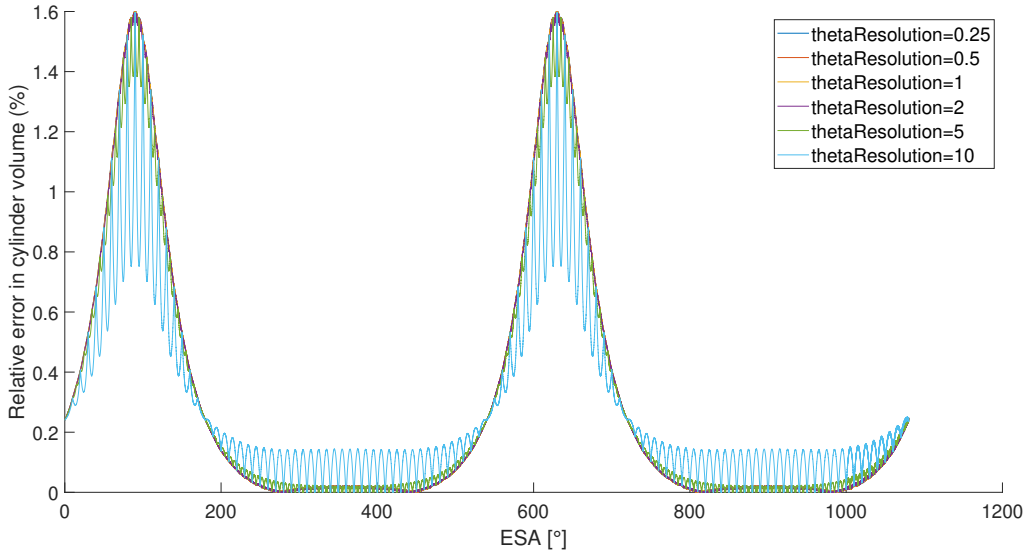
Figure 3.18: Engine displacement law comparison. The detail shows the linear behaviour of the volume curve where it is interpolated between two extreme points.

The results show the presence of an error, which is generated by two contributions, one main component related to the simplifications and assumptions of the model, and one related to the interpolation. The first error is due to the fact that the analytical expression is an approximated formula itself and the code does not implement the precise shapes of (3.3), which takes ϕ into consideration, and (3.8), considering the thickness of the apex seals. Also, as expected, since the engine is relatively small (225cc), the minimum volume V_{min} of the chamber will be very small, and consequently, in the error calculation, a little discrepancy of less than 1cc will weights more. It can be noted that the maximum error of 4.57% (Figure 3.19a) will reduce to 1.6% (Figure 3.19b) if, instead of the catalogue geometrical radius (69.5 mm), the radius R' (71 mm), introduced in 3.1, is used for the calculation of ϕ_{max} (3.11). This substitution can lead to a reduction in the error because it is more consistent with the assumption made in the final part of 3.1.2, according to which the apex seal openings of the rotor shape are not considered. The second type of error is the interpolation error[4], which is created by the fact that the volume curve obtained by the software is defined only in certain discrete steps, identified by the parameter `thetaResolution`, while the intermediate positions are interpolated by the software. Figure 3.20 shows that this interpolation error is negligible and approach 0 for small values of `thetaResolution`.

[4]This error is responsible for the scattering in Figure 3.19a and Figure 3.19b.



(a) Error for ϕ_{max} calculated with R.



(b) Error for ϕ_{max} calculated with R'.

Figure 3.19: These pictures illustrate the deviation from the theoretical volume law, both for the approximation error and the interpolation error.

The second type of error is the interpolation error, which is created by the fact that the volume curve obtained by the software is defined only in certain discrete steps, identified by the parameter `thetaResolution`. Figure 3.20 shows that this interpolation error is negligible and approach 0 for small values of `thetaResolution`.

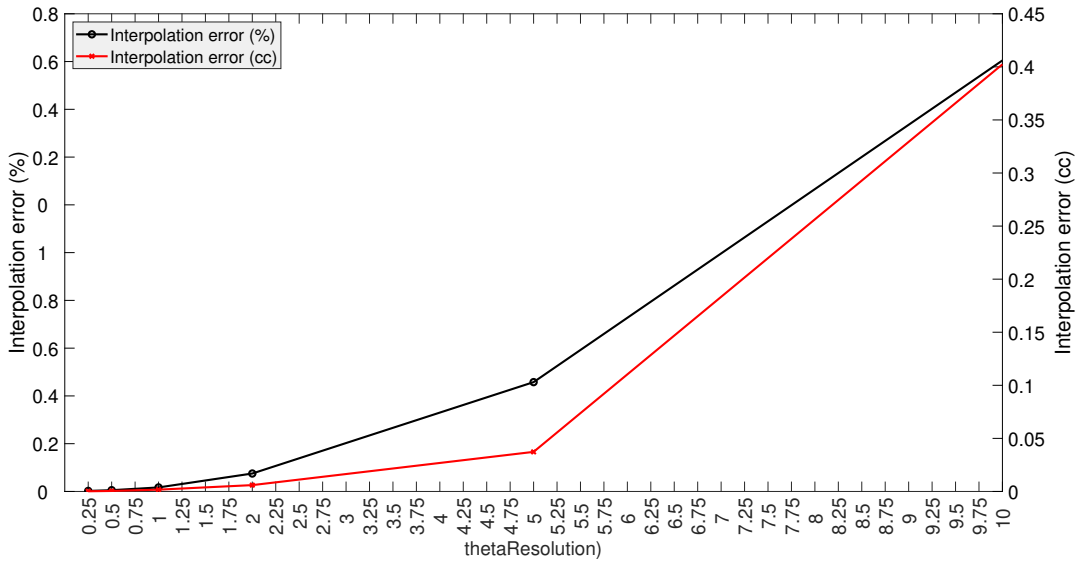


Figure 3.20: Maximum value of the interpolation error with respect to `thetaResolution`.

Overall, it can be stated that the OpenFOAM® model well approximates the analytical engine displacement law and this is confirmed by looking at the CR, which value is 13.46:1 for the model and 12.9:1 (using R) and 13.25:1 (using with $R + a'$) for the two previously mentioned analytically calculated volumes. Clearly, for the reasons stated in 3.1.6, the compression ratio obtained here is bigger than the actual one (9.6:1), due to the missing recess volumes. In conclusion, despite some minor discrepancy for the approximations implemented, it can be stated that the model have a good level of fidelity, that could be improved, when needed, by modifying the equations implemented.

4 | Engine analysis

This chapter is dedicated to the presentation of the simulation results using both the single and three chambers configurations. The simulations were carried out using the compressible, non-reacting flow solver `coldEngineDyMFoam` (it features also spray modeling) and are focused on a first mesh sensitivity analysis, with different configurations, for what concerns the duct lengths, and subsequently, a comparison between the single and three chambers configuration.
For this thesis work a large amount of time and efforts was spent setting up the simulation parameters in order to reach convergence of the model, and this goal was finally reached making some adjustments with respect to the initial setup. Consequently, a section explaining the arrangement and the assumptions of the model will also be presented. In addition, as it will be shown, the simulations made were really computational demanding and time consuming and for this reason the modeling of the spray and combustion remains an off-topic for the present work.

4.1. Case setup

Before the discussion about the case setup it can be useful to have a full overview of its organization[1]:

[1]Some of these dictionaries and files are related to the mesh generation and are already been presented in the previous chapter so they will not be covered again here.

```
Folder with chambers and ducts cases/
├── IntakeExhaustpipes/
├── meshParameters/
├── wankelGeometry/
├── motored/
    ├── constant/
        ├── polyMesh/
        ├── thermophysicalProperties
        ├── momentumTransport
        ├── engineGeometry
        ├── g
        ├── dynamicMeshDict
        ├── combustionProperties
        ├── wankelBlocks
        ├── wankelPolyLines
        ├── wankelPoints
    ├── system/
        ├── blockMeshDict/
        ├── controlDict/
        ├── createBafflesDict/
        ├── decomposeParDict/
        ├── fvSchemes/
        ├── fvSolution/
        ├── setFieldsDict/
        ├── topoSetDict/
    ├── -253/
        ├── alphat/
        ├── CO2/
        ├── epsilon/
        ├── fuel/
        ├── H2O/
        ├── k/
        ├── N2/
        ├── nut/
        ├── O2/
        ├── p/
        ├── T/
        ├── U/
        ├── .../
```

As it can be seen the main folder of the case contains both the duct case, namely the `IntakeExhaustpipes`, and the rotary chambers one, namely the `motored`. It is worth mentioning that the modeling of the combustion physics inside the chamber, meaning the spray and the heat model is an off-topic of the present work, so, to obtain the simulation results and verify their consistency, a combustion-like situation was assumed in every

chamber right before the EPO and the exhaust stroke. To clarify, when each one of the chambers reach the EPO position, that corresponds to a θ angle of -253, the fields inside it are initialized as the combustion occurred in that specific cycle. In particular, at that time step, the fields of pressure, temperature, velocity and chemical species are initialized with the `seFields` using some values that are consistent with other simulations carried out by 1-D approaches and scientific papers regarding the AIE 225CS. Clearly, this approach is not accurate and it may not yield highly reliable physical results, nonetheless, it serves as an initial study since it has allowed for the analysis of how the lengths of the ducts and the number of chambers influence the gas dynamics of the engine under the same initial conditions. In the following subsection a detailed explanation of the setup is presented, highlighting the choices made to obtain a good compromise between accuracy and computational time.

4.1.1. Mesh setup

As it will be explained in detail later, the simulations made for this work were very time consuming, since many revolutions were needed for each simulations to reach convergence of the results. For this reason the mesh simulated was coarse and the values inserted in the `meshParameters` and `wankelEngine` are the following:

As it can be seen the entries related to the `snappyHexMesh` (to simulate the rotor recesses and the real geometry of the pipes) and the spark plugs are not reported, since they were not utilized. In the following sections many configurations with different duct lengths were used and, for this reason, the `nzIntake` and the `nzExhaust` values are chosen consistently with their length to maintain a constant aspect ratio of the duct cells between the various configurations: in this case for the 30 cm duct the number of cells are set equal to 30, while for the 40 cm duct it is equal to 80. The diameter of the cylindrical ducts instead was chosen to obtain two equivalent areas equal to the real engine ones, which has a rectangular-like shape.
Due to the Wankel engine geometry, maintaining low values of mesh non-orthogonality and skewness is not possible and specific choices in the setup must be taken into account to limit the errors. The command `checkMesh` on the initial engine position (-253 ESA) gives the following output: However, the values of non-orthogonality and aspect ratio changes during the rotation. In fact, if their values are plotted for the complete cycle of 1080 ESA, higher values will be obtained (figure 4.1).

meshParameters	
Parameter	**Value**
Nx	10
Ny	10
Nz	50
thetaResolution	1
pLineResolution	100
thetaIntakeDeg	52 degrees (counterclockwise)
thetaExhaustDeg	125 degrees (counterclockwise)
dIntake	0.0245 m
dExhaust	0.026 m
lIntake	300-600 mm
lExhaust	300-600 mm
nxyIntake	6
nxyExhaust	6
nzIntake	60-120
nzExhaust	60-120
intakeOffset	0
exhaustOffset	0
sqRatioIntake	0.7
sqRatioExhaust	0.7
mergeMesh	true
caseToMerge	intakeExhaustPipes

wankelGeometry	
Parameter	**Value**
thickness	51.94 mm
rotorRadius	71 mm
eccentricity	11.6 mm
clearance	0.5 mm

Table 4.1: Mesh parameters and Wankel geometry

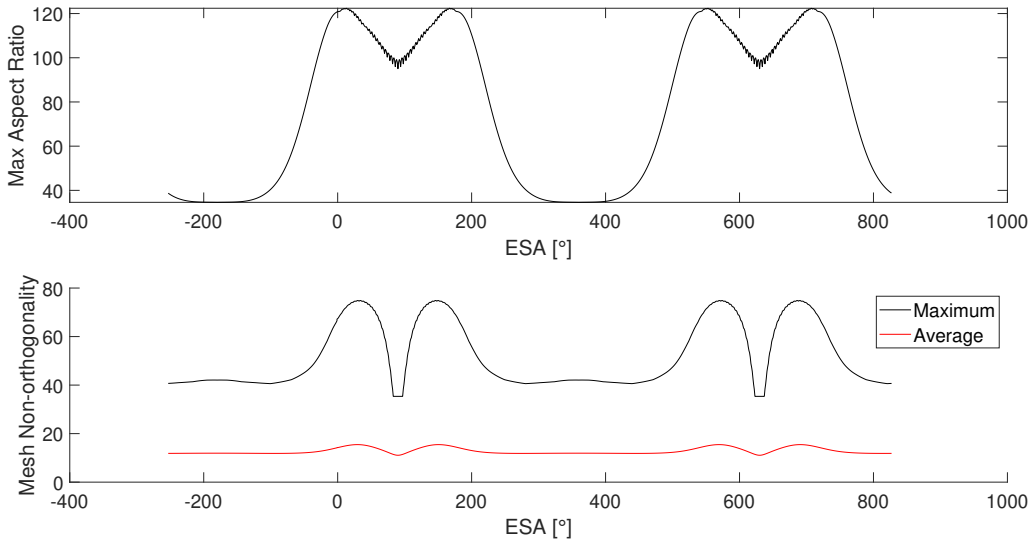


Figure 4.1: Mesh non-orthogonality and aspect ratio for the full cycle.

Parameter	Value
Number of points	29717
Number of faces	83560
Number of internal faces	76900
Number of cells	26600
Number of regions	3
Boundary openness	(-4.87789e-16, -5.7145e-17, -1.91533e-16)
Max cell openness	7.20692e-16
Max aspect ratio	38.7166
Min face area	1.32885e-16 m^2
Max face area	2.35093e-05 m^2
Total volume	0.000553856 m^3
Max non-orthogonality	40.6979
Average non-orthogonality	11.8557
Max skewness	2.10824

Table 4.2: CheckMesh summary

4.1.2. Boundary and initial conditions

The domain of the simulation is composed by the patches listed in Figure 4.2, where the three-chamber configuration was chosen to better highlight the various surfaces (the seals patches are not visible but they are present at each point where one chamber is in contact with the adjacent one).

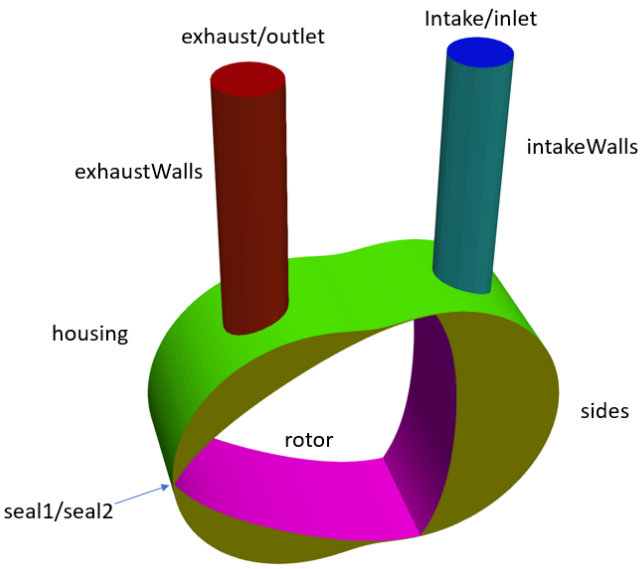


Figure 4.2: Patches of the simulation.

Boundary conditions are implemented in the files present in the `-253` folder, which is the starting ESA of the simulation, and corresponds to the EPO. Table 4.3 will present the various patches and the boundary conditions applied to them for pressure, velocity and temperature.

Table 4.3: Boundary conditions for p, U, and T

Boundary	**p [Pa]**	**U [m/s]**	**T [K]**
intakeWalls	zeroGradient	fixedValue (0 0 0)	fixedValue (500)
exhaustWalls	zeroGradient	fixedValue (0 0 0)	fixedValue (500)
seal1	zeroGradient	movingWallVelocity (0 0 0)	fixedValue (500)
seal2	zeroGradient	movingWallVelocity (0 0 0)	fixedValue (500)
rotor	zeroGradient	movingWallVelocity (0 0 0)	fixedValue (500)
sides	zeroGradient	fixedValue (0 0 0)	fixedValue (500)
intake	totalPressure (101325)	pressureInletOutletVelocity (0 0 0)	inletOutlet (300)
exhaust	totalPressure (101325)	pressureInletOutletVelocity (0 0 0)	inletOutlet (600)
housing	zeroGradient	fixedValue (0 0 0)	fixedValue (500)
ACMI1_couple	cyclicACMI (100000)	cyclicACMI (0 0 0)	cyclicACMI (600)
ACMI1_blockage	zeroGradient	fixedValue (0 0 0)	fixedValue (500)
ACMI2_couple	cyclicACMI (100000)	cyclicACMI (0 0 0)	cyclicACMI (600)
ACMI2_blockage	zeroGradient	fixedValue (0 0 0)	fixedValue (500)

For the pressure, as it can be seen, the `zeroGradient` boundary condition is used for all the patches that are also defined as walls, implying that the pressure gradient normal to the boundary is zero, while the `totalPressure` boundary condition is used to impose the ambient pressure at the inlet and outlet patches, which is useful for compressible flow simulations because it accounts for both static and dynamic pressure contributions, allowing the solver to more accurately model the complex coupling between pressure, velocity, and temperature fields. Concerning the velocity, instead, the `fixedValue` is used for walls with no-slip conditions (velocity = 0), while the `movingWallVelocity` is used for moving walls (e.g., seals and rotor). It sets the velocity of the wall as the boundary condition. For intake and exhaust patches, the `pressureInletOutletVelocity` is used. This boundary condition is used for inlet and outlet boundaries where the flow direction is not known

a priori. The velocity is calculated based on the pressure field, and it allows the flow to enter or exit the domain depending on the pressure distribution. Finally, the already mentioned ACMI patches are set with cyclicACMI condition. It allows for the correct interpolation of values across the interface, taking into account possible relative motion between the two cyclic patches. For temperature, some reasonable initial values are assumed and, together with the `fixedValue` applied at walls, the `inletOutlet` boundary condition is applied at the inlet and outlet patches. The temperature is fixed if the flow is entering the domain, and the gradient is zero if the flow is exiting the domain. It is useful for cases where the flow direction is not known a priori.
The remaining boundary conditions are related to the chemical species initialization. About this topic, as anticipated previously, the fields were initialized for the whole simulation domain using the `setFields` utility. This command modifies the `internalField` value of the various `cellZones` in the simulation with the ones specified by the user. In this setup, the engine draws in air (consisting of $21\% O_2$ and $79\% N_2$) and iso-octane as fuel, and expels exhaust gases as if a stoichiometric reaction took place. The equation of the reaction is:

$$C_8H_{18} + 12.5\ (O_2 + 3.76\ N_2) \rightarrow 8\ CO_2 + 9\ H_2O + 47\ N_2 \tag{4.1}$$

It must be said that the initial simulations used kerosene as fuel and assumed a reaction with an equivalence ratio ϕ of 1.2[2], but when the convergence problems arose, it was suspected that an incorrect setup of the chemical species could be the cause. As a result, a simpler configuration was chosen. In fact, iso-octane was selected because it is a good reference fluid for gasoline combustion simulations. The properties of the fuel have been specified in the `constant/thermoPhysicalProperties` file.
From the reaction (4.1) the mass fractions of the fresh charge and the exhaust gases can be calculated and imposed through the `setfieldsDict` file:

```
defaultFieldValues
(
    ...
);

regions
(
    zoneToCell
```

[2]To simulate a rich mixture and incomplete combustion, which is likely to occur for a heavy fuel like kerosene.

```
{
    name rotorCells1;
    fieldValues
    (
        volScalarFieldValue p 5.6e5
        volVectorFieldValue U (0 0 0)
        volScalarFieldValue T 1400
        volScalarFieldValue CO2 0.19227985458836
        volScalarFieldValue H2O 0.0884924330775975
        volScalarFieldValue N2 0.719227712334043
        volScalarFieldValue fuel 0
        volScalarFieldValue O2 0
    );
}
zoneToCell
{
    name intakeCells;
    fieldValues
    (
        ...
    );
]
zoneToCell
{
    name exhaustCells;
    fieldValues
    (
        ...
    );
]
zoneToCell
{
    name sparkCells;
    fieldValues
    (
        ...
    );
```

```
    ]
// ...
)
```

The `regions` block is used to define a set of regions (cell zones) and their corresponding initial field values. Each region is specified using a `zoneToCell` block, which contains the name of the cell zone and a list of field values to be assigned to that region (see table 4.4).

Cell Zone	**p (Pa)**	**U (m/s)**	**T (K)**	**CO2**	**H2O**	**N2**	**fuel**
rotorCells	5.6e5	(0, 0, 0)	1400	0.1923	0.0885	0.7192	0.0
intakeCells	1e5	(0, 0, 0)	300	0.0	0.0	0.7192	0.0623
exhaustCells	1e5	(0, 0, 0)	600	0.1923	0.0885	0.7192	0.0
sparkCells	Not used						

Table 4.4: Field values for different cell zones

The temperature and pressure values were set based on the informations found in literature and experiments and subsequently, for each simulation tuned to ensure mass conservation. To clarify, the temperature value was kept 1400 K, while the pressure value was selected so that the value of mass inside the cylinder when the fields are initialized, at -253, was equal to the one that was trapped inside the chamber from the previous cycle. It can be seen that the intake and exhaust are initialized with atmospheric pressure, this implies that the simulation will need many revolutions to reach convergence. In particular it was found that 4 complete revolutions (1080x4 ESA) are needed to obtain the same outputs as the previous cycle and, for this reason, the simulation results were considered reliable from the third revolution (the one going from 1907 to 2987 ESA). Additionally, it is important that the mass fractions of the initialization are consistent with those imposed in the boundary conditions, especially on the patches of inlet and outlet. The initial conditions for chemical species are shown in table 4.5, while the mixture fraction (Z), its variance (Z2) and the default mass fraction for the other species that may be produced are shown in table 4.6.

Boundary	**CO2** [%]	**H2O** [%]	**N2** [%]	**O2** [%]	**Fuel** [%]
intakeWalls	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
exhaustWalls	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
seal1	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
seal2	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
rotor	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
sides	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
intake	inletOutlet (0)	inletOutlet (0)	inletOutlet (0.719..)	inletOutlet (0.218..)	inletOutlet (0.062..)
exhaust	inletOutlet (0.192..)	inletOutlet (0.088..)	inletOutlet (0.719..)	inletOutlet (0)	inletOutlet (0)
housing	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
ACMI1_couple	cyclicACMI	cyclicACMI	cyclicACMI	cyclicACMI	cyclicACMI (0.0123013)
ACMI1_blockage	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient
ACMI2_couple	cyclicACMI	cyclicACMI	cyclicACMI	cyclicACMI	cyclicACMI (0.0123013)
ACMI2_blockage	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient

Table 4.5: Boundary conditions for chemical species

Boundary	Ydefault	Z	Z2
intakeWalls	zeroGradient	zeroGradient	zeroGradient
exhaustWalls	zeroGradient	zeroGradient	zeroGradient
seal1	zeroGradient	zeroGradient	zeroGradient
seal2	zeroGradient	zeroGradient	zeroGradient
rotor	zeroGradient	zeroGradient	zeroGradient
sides	zeroGradient	zeroGradient	zeroGradient
intake	inletOutlet (uniform 0)	inletOutlet (uniform 0)	inletOutlet (uniform 0)
exhaust	inletOutlet (uniform 0)	inletOutlet (uniform 0)	inletOutlet (uniform 0)
housing	zeroGradient	zeroGradient	zeroGradient
ACMI1_couple	cyclicACMI (uniform 0)	cyclicACMI (uniform 0)	cyclicACMI (uniform 0)
ACMI1_blockage	zeroGradient	zeroGradient	zeroGradient
ACMI2_couple	cyclicACMI (uniform 0)	cyclicACMI (uniform 0)	cyclicACMI (uniform 0)
ACMI2_blockage	zeroGradient	zeroGradient	zeroGradient

Table 4.6: Boundary conditions for default mass fraction of additional species, mixture fraction, and mixture fraction variation.

4.1.3. Turbulence model

The model used is k-ε, which is valid for y+ values ranging from 30 to 300. However, in Wankel, and generally in internal combustion engines, due to aspects of geometrical complexity it is almost never possible to realize a consistent boundary layer and reconducting to a steady-state simulation. The $k-\epsilon$ was chosen for its versatility and the possibility to use wall functions to approximate the effects of the sublayers near the walls. The model of the turbulent must be selected in the `momentumTransport` file and the wall functions are specified in the initial folder of the case. The boundary conditions for thermal diffusivity, turbulent kinematic viscosity and turbulent kinetic energy are hereby reported.

α_T		
Boundary	Type	Parameters
intakeWalls	AngelbergerWallFunction	C1=2.075, C2=3.9, yPlusSwitch=13.2, Cmu=0.09, kappa=0.41, E=9.8, cHeat=1, maxYPlus=1e+15, value=uniform 0
exhaustWalls	AngelbergerWallFunction	(same as intakeWalls)
seal1	AngelbergerWallFunction	(same as intakeWalls)
seal2	AngelbergerWallFunction	(same as intakeWalls)
rotor	AngelbergerWallFunction	(same as intakeWalls)
sides	AngelbergerWallFunction	(same as intakeWalls)
intake	calculated	value=uniform 0
exhaust	calculated	value=uniform 0
housing	AngelbergerWallFunction	(same as intakeWalls)
ACMI1_couple	cyclicACMI	value=uniform 0
ACMI1_blockage	AngelbergerWallFunction	(same as intakeWalls)
ACMI2_couple	cyclicACMI	value=uniform 0
ACMI2_blockage	AngelbergerWallFunction	(same as intakeWalls)

Table 4.7: Initial conditions for the thermal diffusivity.

nut		
Boundary	Type	Parameters
intakeWalls	nutkWallFunction	value=uniform 0
exhaustWalls	nutkWallFunction	(same as intakeWalls)
seal1	nutkWallFunction	(same as intakeWalls)
seal2	nutkWallFunction	(same as intakeWalls)
rotor	nutkWallFunction	(same as intakeWalls)
sides	nutkWallFunction	(same as intakeWalls)
intake	calculated	value=uniform 0
exhaust	calculated	value=uniform 0
housing	nutkWallFunction	(same as intakeWalls)
ACMI1_couple	cyclicACMI	value=uniform 0
ACMI1_blockage	nutkWallFunction	(same as intakeWalls)
ACMI2_couple	cyclicACMI	value=uniform 0
ACMI2_blockage	nutkWallFunction	(same as intakeWalls)

Table 4.8: Initial conditions for the turbulent kinematic viscosity

k		
Boundary	Type	Parameters
intakeWalls	kqRWallFunction	Cmu=0.09, kappa=0.41, E=9.8, value=uniform 1
exhaustWalls	kqRWallFunction	(same as intakeWalls)
seal1	kqRWallFunction	(same as intakeWalls)
seal2	kqRWallFunction	(same as intakeWalls)
rotor	kqRWallFunction	(same as intakeWalls)
sides	kqRWallFunction	(same as intakeWalls)
intake	inletOutlet	inletValue=uniform 1, value=uniform 1
exhaust	inletOutlet	(same as intake)
housing	kqRWallFunction	(same as intakeWalls)
ACMI1_couple	cyclicACMI	value=uniform 1
ACMI1_blockage	kqRWallFunction	(same as intakeWalls)
ACMI2_couple	cyclicACMI	value=uniform 1
ACMI2_blockage	kqRWallFunction	(same as intakeWalls)

Table 4.9: Initial conditions for the turbulent kinetic energy

ϵ		
Boundary	Type	Parameters
intakeWalls	epsilonWallFunction	Cmu=0.09, kappa=0.41, E=9.8, value=$internalField
exhaustWalls	epsilonWallFunction	(same as intakeWalls)
seal1	epsilonWallFunction	(same as intakeWalls)
seal2	epsilonWallFunction	(same as intakeWalls)
rotor	epsilonWallFunction	(same as intakeWalls)
sides	epsilonWallFunction	(same as intakeWalls)
intake	inletOutlet	inletValue=uniform 6.32, value=uniform 6.32
exhaust	inletOutlet	(same as intake)
housing	epsilonWallFunction	(same as intakeWalls)
ACMI1_couple	cyclicACMI	value=uniform 6.32
ACMI1_blockage	epsilonWallFunction	(same as intakeWalls)
ACMI2_couple	cyclicACMI	value=uniform 6.32
ACMI2_blockage	epsilonWallFunction	(same as intakeWalls)

Table 4.10: Initial conditions for the rate of dissipation of turbulent kinetic energy.

4.1.4. Solver and numerical setup

The solver used for the simulations is the already mentioned `coldEngineDyMFoam`, a compressible flow solver for engine meshes undergoing deformation or topological changes with the possibility to handle spray. All the simulations are performed at 7500 rpm, which is from the catalogue the point where the peak power is reached. The rpm must be set up in the `engineGeometry` dictionary, which contains also the data about the mesh (the file `meshParameters` is included), the swirl characteristic of the engine, and the correction for the flux mentioned in 3.4.2.

As explained before, the solving equations are being discretized in order to solve the system. The numerical schemes are specified in the files `fvSchemes` and the solution setup in `fvSolutions`. The `fvSchemes` handles the discretization of gradient, divergence and laplacian terms and is a crucial file for the stability of the simulation. In fact the setup hereby presented will be focused on this aspect, rather than accuracy, because of the numerous stability problems encountered. As best practice, improving the mesh accuracy, moving towards a fine mesh setup and using higher order schemes can improve the accuracy of the results.

Starting with the time derivative schemes, Euler was selected. It is a first-order accurate

explicit time-stepping scheme:

```
ddtSchemes
{
    default Euler;
}
```

The gradient schemes chosen is Gauss linear, which computes gradients using a Gauss linear method. It computes the gradient by averaging the values at the cell faces, which are in turn obtained by linear interpolation of the neighboring cell center values. However, the Gauss linear scheme can sometimes produce unphysical results, particularly in regions where the field has large gradients, which is the case for Wankel engines. In these cases, the linear interpolation may introduce overshooting or undershooting, where the interpolated values exceed the actual maximum or minimum values of the neighboring cells. This can lead to numerical instabilities and unreliable simulation results. In Wankel engine simulations, there are critical regions where the velocity field presents high gradients, for example when the chamber volume is minimum or when the ports are closing. In fact, in this work, the ports are modeled as cylindrical and this cause a non optimized valve opening profile, which is characterized by very small areas when the ducts are initially closing or opening. Instead, the real shape of the ports is usually rectangular-like, allowing a smoother gradient for the fields velocity due to larger sections. To address this issue, the `faceLimited Gauss linear` scheme adds a limiter to the standard Gauss linear method. The limiter constrains the interpolated values at cell faces based on the values of neighboring cells. By doing so, it prevents overshooting and undershooting while still allowing accurate gradient computation in regions with smooth variations. The `faceLimited Gauss` is usually preferred and is usually more accurate for sharp gradients with respect to the `cellLimited Gauss`, but it can be less stable.

```
ddtSchemes
{
   gradSchemes
{
    default         Gauss linear;
    grad(U)         faceLimited Gauss linear 1.0;
}
}
```

Divergence schemes are used to discretize the divergence terms in the PDEs. The choice of the scheme can greatly affect the overall accuracy, stability, and convergence of the

simulation. For example, upwind schemes are typically more stable and robust, while limited linear schemes provide higher-order accuracy but may require finer mesh resolution. Since the preliminary mesh used for this work is coarse and problems on simulation stability occurred, many quantities have been discretized using the "upwind" scheme, e.g. turbulence and chemical species. For what concern the turbolence, the k and ε are very susceptible to numerical oscillations and setting them to "upwind" it can be assured that they follows the flux. Also, the interest is more on the local generation of these quantities more than their transport, since they are geneated in correspondance of high velocity gradients but tends to dissipate quickly. Further enhancement of the mesh can potentially allow to move towards more accurate schemes without compromising the stability.

```
divSchemes
{

    default          none;

    div(phi,U)       Gauss linearUpwind grad(U);
    div(phi,Z2) Gauss limitedLinear 1;
    div(phi,YNOPollutant) Gauss limitedLinear 1;
    div(phi,YNOPollutantCFD)  Gauss limitedLinear 1;
    div(phi,YNOCFDRifSource) Gauss limitedLinear 1;



    div(phi,k)       Gauss upwind;
    div(phi,epsilon) Gauss upwind;

    div(phiU,p)      Gauss limitedLinear 1;
    div(phid,p)      Gauss limitedLinear 1;
    div(meshPhi,p)   Gauss limitedLinear 1;
    div(phi,Yi_h)    Gauss multivariateSelection
    {
       Z upwind;
       Z2 upwind;
       h limitedLinear 1.0;
       O2 upwind;
       N2 upwind;
       CO2 upwind;
```

```
        H2O upwind;
        fuel upwind;
        H2 upwind;
        CO upwind;
    };
    div(phi,K) Gauss upwind;
    div((phi|interpolate(rho)),p) Gauss limitedLinear 1;

    div((phi|interpolate(rho)),cNp) Gauss limitedLinear 1;
    div((phi|interpolate(rho)),rhosFv) Gauss limitedLinear 1;

    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div((muEff*dev2(T(grad(U))))       Gauss linear;
}
```

The Laplacian schemes used for discretizing the Laplacian operator is Gauss linear limited with a 0.33 coefficient, which is a limited version of Gauss linear for better numerical stability. It is a second order accurate discretization with a limiter, that is introduced again to avoid overshooting and undershooting. The limiter is set to 0.33, which means that the calculated gradient will be limited to 33% of the maximum allowable gradient between neighboring cells.

```
    laplacianSchemes
{
    default                 Gauss linear limited 0.33;
}
```

The interpolationSchemes dictionary specifies the interpolation schemes used for various fields in the simulation. Interpolation is the process of estimating the values of a field variable between discrete points (e.g., cell centers) in the mesh and for simplicity and computational efficiency is set to linear in this case.

```
    interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
```

Due to the geometry of Wankel engine and the resulting high values of skewness and non-orthogonality of the mesh it was chosen to adopt as default the "corrected" entry

for the `snGradSchemes` (Surface-normal Gradient Schemes), which in turn, is designed to handle these problems by introducing the non-orthogonal correction, leading to a more representative and stable solution.

```
    snGradSchemes
{
    default         corrected;
}
```

The `fluxrequired` entry is used to specify whether the solver needs to calculate the flux of a particular field variable across the mesh boundaries. This information is needed to ensure that the boundary conditions are correctly set up for the simulation. It was activated only for the pressure and the pressure correction, to ensure accurate pressure-velocity coupling in the calculations.

```
   fluxRequired
{
    default  no;
    p;
    pcorr;
}
```

The `wallDist` is needed for the calculation of the wall distance, which is needed for turbulence models and wall functions. The method `meshWaves` is particularly suitable for large and complex mesh geometries like the Wankel ones, as it can handle both structured and unstructured meshes and provide accurate wall distance calculations without excessive computational cost.

```
    wallDist
{
    method meshWave;
}
```

For the choice of the solvers for the solution of equations, meaning the `fvSolution`, instead, the following setup was chosen. For the section related to the solvers, which handles the settings for solving linear equations related to different variables, the PBiCGStab solver was chosen. The PBiCGStab algorithm is an improvement over the original BiCG (Biconjugate Gradient) method, which can suffer from numerical instability issues.
PBiCGStab adds a stabilization step to mitigate these issues, resulting in a more stable and reliable solver. DILU (Diagonal Incomplete LU) is a preconditioner used to improve

the convergence of the PBiCGStab solver. The solver is often preconditioned to improve the convergence rate. Preconditioning involves transforming the original linear system into another system with the same solution but better numerical properties, making it easier for the iterative solver to converge. The solver will stop iterating once the residual falls below 1e-19, ensuring a high level of accuracy in the solution.

```
    solvers
{
    "(p|rho)"
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-19;
        relTol          0.;
    }

    "(p|rho)Final"
    {
        $p;
        relTol          0;
    }

    "(U|h|hu|enthalpy|c|Yi|k|epsilon)"
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-19;
        relTol          0.;
    }

    "(U|h|hu|enthalpy|c|Yi|k|epsilon)Final"
    {
        $U;
        relTol          0;
    }

    "(Yi|YNOPollutant)"
```

```
    {
        $hFinal;
    }
}
```

Finally, the solver uses PIMPLE algorithm, with the `transonic` option set to on. It is activated to allow the solver to use specific algorithms and techniques to better capture the changes in flow properties and behavior associated with the transition between subsonic and supersonic flow regimes. In addition, for the `nNonOrthogonalCorrectors` the value of iteration is set to 6, which helps to improve the solution accuracy in case of highly skewed and distorted meshes, despite increasing also computational cost. `nOuterCorrectors` and `nCorrectors` defines the the number of outer iterations for the PIMPLE algorithm and the PISO corrector loops within each outer iteration, respectively. They are not increased too much to keep smaller the computational cost.

```
PIMPLE
{
    transonic on;
    momentumPredictor off;
    nOuterCorrectors 3;
    nCorrectors     1;
    nNonOrthogonalCorrectors 6;
}
```

4.2. Mesh sensitivity analysis

This section has the goal to present the effect of the ducts lengths variation on the Wankel engine performance. The simulations were carried out with the single chamber configuration at 7500 rpm, which is the rpm value of interest for the Wankel engine, which usually is designed and optimized for a certain rpm value. The lengths of the pipes have been varied from 300 to 600 both for intake and exhaust and their effect on the volumetric efficiency λ_V was calculated. For these simulations the timestep in the `controlDict` file was kept pretty high (0.125 s) to reduce the computational time. This causes the Courant number to reach values near 10 at some point in the simulation (figure 4.3), however, despite this usually means low accuracy for the PIMPLE algorithm, it was demonstrated[3] that the results were consistent with those obtained with a smaller timestep and a Courant number

[3]For a partial number of configurations, tests were taken both with Courant number equal to 1 and Courant number equal to 10.

lower than 1. For this reason it was assumed that these values of timestep and Courant number are acceptable for the preliminary sensitivity analysis on the ducts lengths.

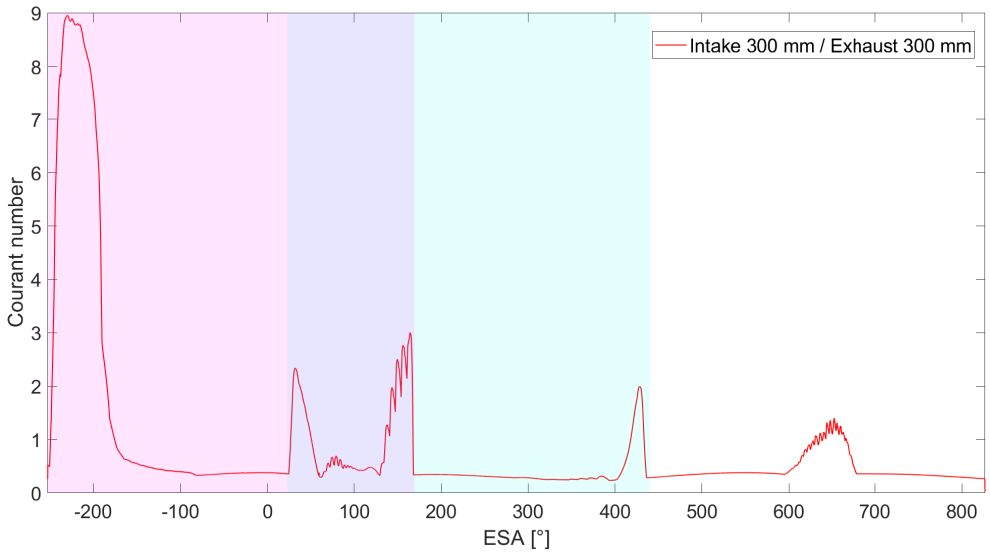


Figure 4.3: Courant number for 0.125 s time step. Simulation with 300 mm intake and 400 mm exhaust was taken as example. The light blue patch represents the intake phase ESA interval, while the red one represents the exhaust one.

To obtain reliable results, without simulating the combustion, the following procedure has been implemented for each configuration:

- Initialize the pressure inside the chamber and the pipes with the conditions shown in table 4.4.
- Run the simulations for a certain number of full cycles (1080 ESA each one) until, looking at the plot of the mass inside the cylinder, the trend of the mass of that cycle is exactly equal to the previous one, meaning that the results of the simulations are stable and settled. In fact, since it has been assumed atmospheric pressure inside the pipes, the simulation needs some time to reproduce the real conditions inside the ducts and represent the real operating condition of the engine at those rpms.
- Identify the mass value inside the cylinder when the simulation achieves stable results. Then, while maintaining a constant temperature of 1400 K, adjust the initial pressure value in the chamber using the `setFields` command. This adjustment should ensure that the resulting mass after initialization matches the mass trapped inside the chamber during the previous cycle.

- Run again the simulation with the new value of pressure until the stability of results is obtained. This time the simulation will reach convergence of the results faster than the case with the non-optimized pressure. This is evident looking at figure 4.4, where convergence is reached for the second cycle, in fact the following cycles shows exactly the same trend for mass.

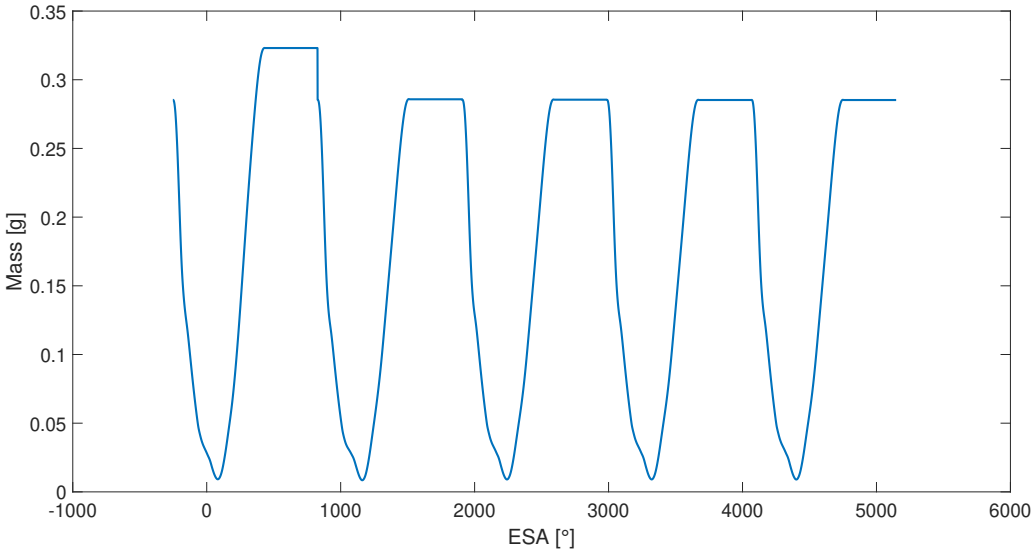


Figure 4.4: Resultant trapped mass for each cycle for the 500 cm intake and 500 cm exhaust configuration.

Generally the cycle from 1907 ESA to 2987 ESA (third cycle) is considered reliable for all the configurations and used for the calculation of the volumetric efficiency. The previous procedure implies that each configuration will have its own pressure which ensures mass consistency between the cycles. Table 4.11 shows all the resulting pressures.

	Exhaust [mm]			
Intake [mm]	300	400	500	600
300	5.960	5.790	5.722	5.772
400	5.757	5.726	5.734	5.735
500	5.543	5.655	5.701	5.563
600	5.536	5.571	5.545	5.484

Table 4.11: Pressures [bar] for all the configurations

To better understand the trends of the cycle quantities from 1907 to 2987, these values have been shifted to correspond with the first cycle, ranging from -253 to 827. The

adjusted data is presented below. For clarity, out of the total 16 configurations, it was chosen to represent the thermodynamic quantities only for the configurations with fixed intake at 300 mm (with varying exhaust lengths) and fixed exhaust of 300 (with varying intake). Figure 4.5 illustrates the pressure trends for varying exhaust lengths with fixed intake lengths of 300 mm. The plot analysis demonstrates that altering the exhaust lengths results in a shift of the oscillating pressure peak towards the right. This shift enables improved control over the pressure experienced by the chamber during the IPO. This observation is critical in optimizing the design of the exhaust system for enhanced performance and reliability in practical applications.

Figures 4.7 and 4.8 illustrate the trends of mass inside the chambers for various configurations, highlighting a larger filling for shorter exhaust ducts. These observations offer crucial insights into the impact of varying exhaust and intake lengths on the engine's overall performance, showing that the intake duct has the main role in chamber filling.

As an example to provide an understanding of the temperature inside the chamber, Figure 4.9 is presented.

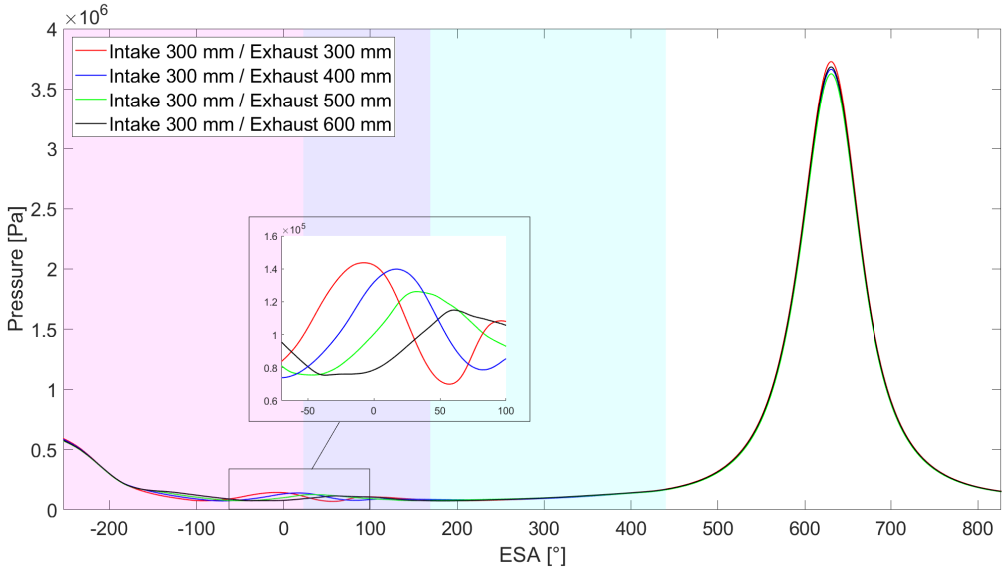


Figure 4.5: Pressure trends varying exhaust lengths for the configuration with 300mm intake.

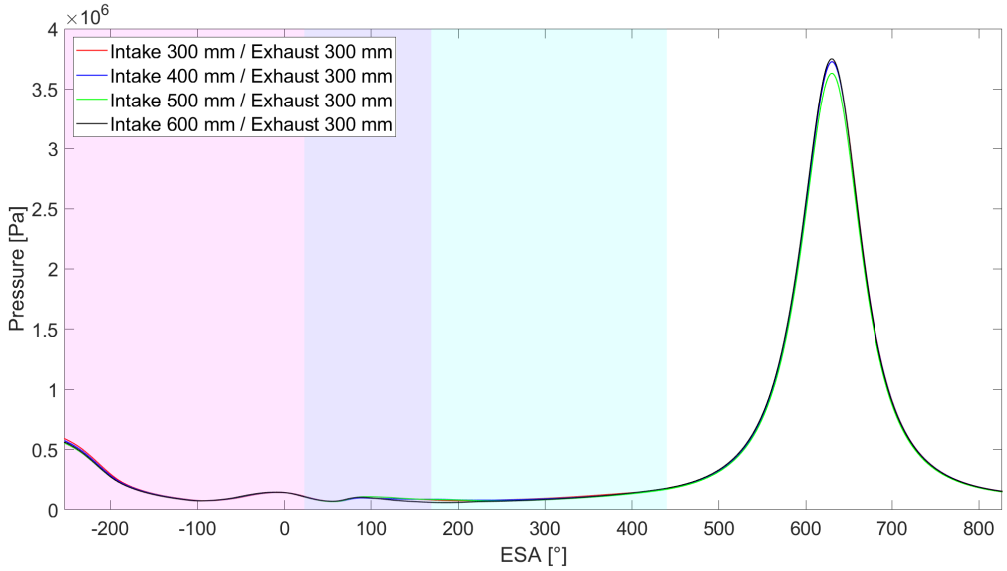


Figure 4.6: Pressure varying intake lengths with fixed exhaust length 300 mm.

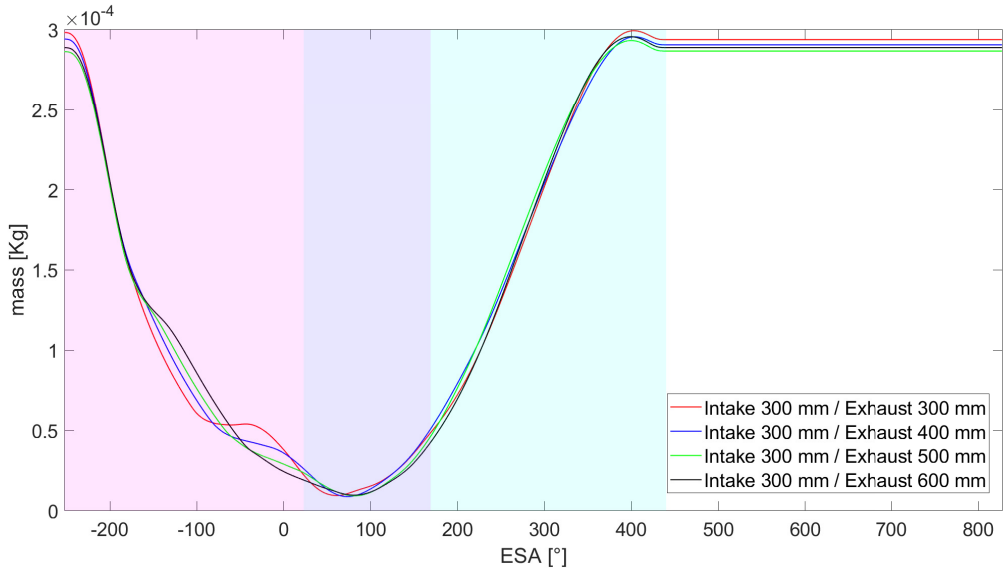


Figure 4.7: Mass value varying exhaust lengths for fixed 300 mm intake.

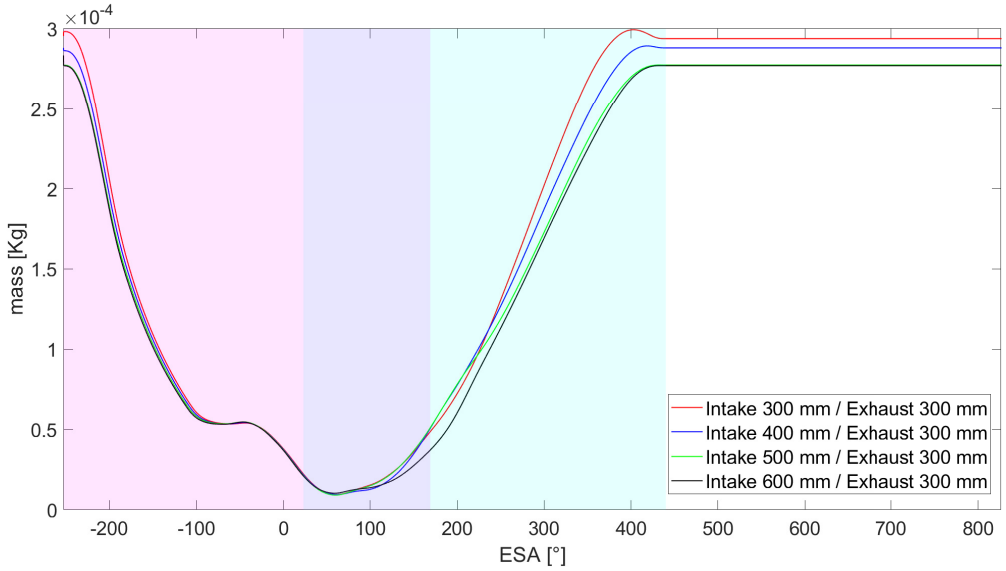


Figure 4.8: Mass value varying intake lengths for fixed 300 mm exhaust.

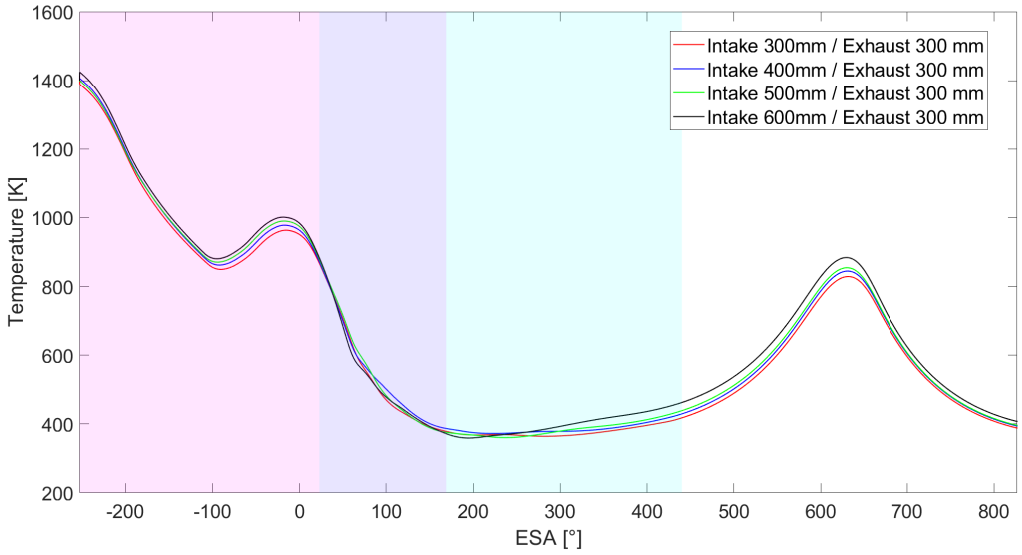


Figure 4.9: Temperature value inside the chamber varying intake length.

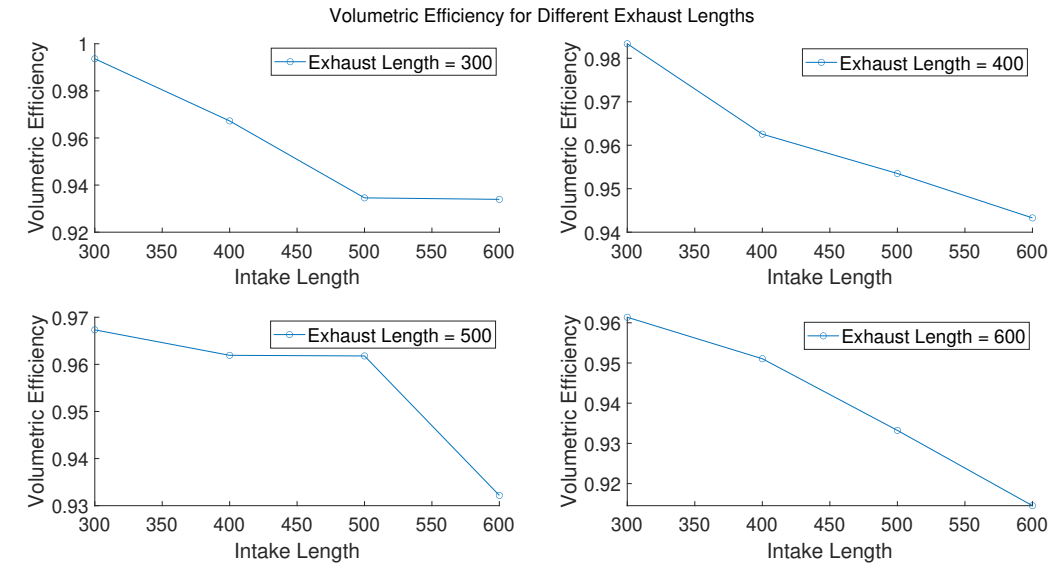
It has been chosen to further analyze the result by looking at the volumetric efficiency to have a better idea of the engine performance with different configurations. This parameter synthetically measures the effectiveness of the global intake process of a given engine. It measures the degree of exploiting the available volume displaced by the pistons to induce a new charge in the chambers.

The volumetric efficiency has been calculated using equation:

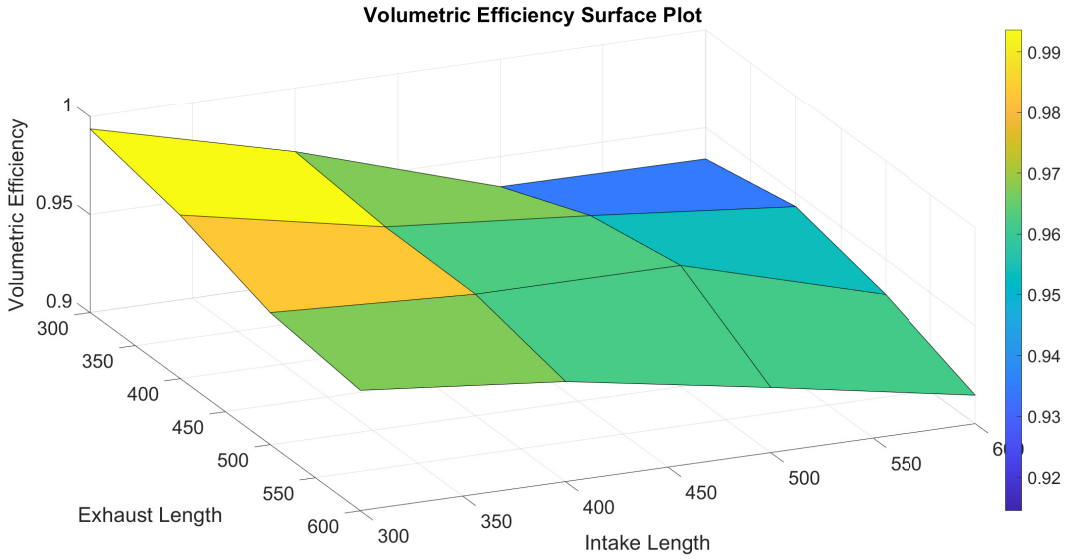
$$\lambda_v = \frac{m_{air}}{\rho_{\text{air}}(V_{\max} - V_{\min})} \tag{4.2}$$

Where ρ_{air} has been calculated at the reference conditions of $p = 1$ bar and $T = 298.15K$. The mass of air has been calculated starting from the O2 mass fraction inside the cylinder. In fact, since the simplified air composition (79% nitrogen and 21% oxygen) is considered, by multiplying $m_{O2}\left(1 + \frac{0.767}{0.233}\right)$, it is possible to obtain the quantity of air trapped into the chamber after all the ports are closed. The volumetric efficiency is predominantly influenced by pressure loss in the intake rather than the exhaust [33]. Therefore, to effectively represent the volumetric efficiency, it has been chosen to vary the intake lengths for each exhaust length configuration.

This approach provides a clearer understanding of the impact of intake lengths on volumetric efficiency. From figure 4.10, it is possible to see how shorter configurations of the intake duct lead to a higher filling of the chamber 4.10a. In particular, upon examining Figure 4.10b, it is evident that the volumetric efficiency coefficient λ_V exhibits a peak for the configuration employing a 300 mm intake pipe coupled with a 300 mm exhaust pipe. This observation implies that further reduction in pipe lengths may potentially reveal an optimal configuration tailored specifically for this particular RPM value. Overall, λ_V exhibits high values, which can be attributed to the elevated RPM and the assumptions made for the model that employs a coarse mesh.



(a)



(b)

Figure 4.10: Volumetric efficiencies as function of ducts length.

4.3. Gas Dynamic comparison

The objective of this section is to present the gas dynamic behavior of the Wankel engine simulations for both single and three-chamber configurations. The simulations were performed with a smaller timestep than in the previous cases to ensure the accuracy of the results. The Courant Number in this case is kept below 1 (Figure 4.11), which is synonym of good accuracy for the PIMPLE algorithm.

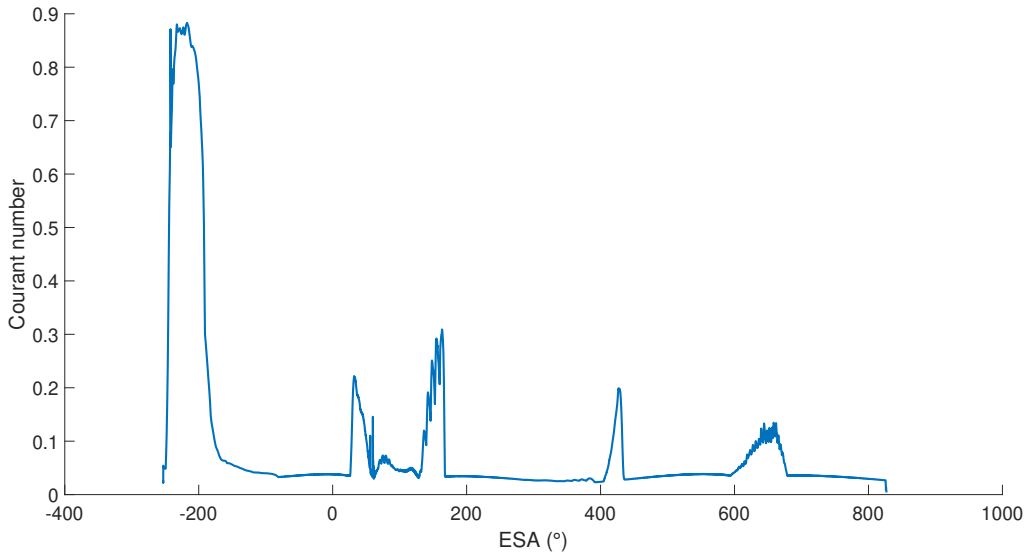


Figure 4.11: Courant number trend for the simulation with single chamber and 0.0125 s timestep.

The configuration chosen for comparison comprises 300 mm intake and 400 mm exhaust pipes. To achieve stable simulation results, the same approach as in the previous section was utilized. All other settings, such as mesh refinement, rpm, and numerical setup, were kept consistent with the earlier cases. To assess the pressure trend within the ducts, pressure probes were placed along their lengths using `functionObjects`, allowing for a more accurate measurement of the pressure variations.
The opening and closing time of the ports are the following:

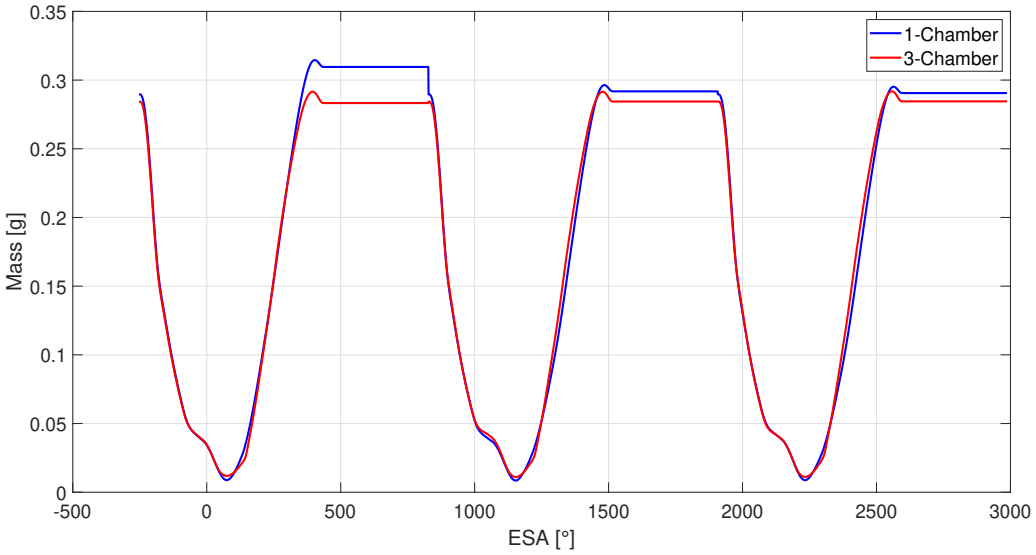
Ports Timing				
	EPO [°]	**EPC [°]**	**IPO [°]**	**IPC [°]**
Chamber 1	-253	168	22	439
Chamber 2	467	-192	742	79
Chamber 3	107	528	382	799

Table 4.12: Ports timing for different chambers.

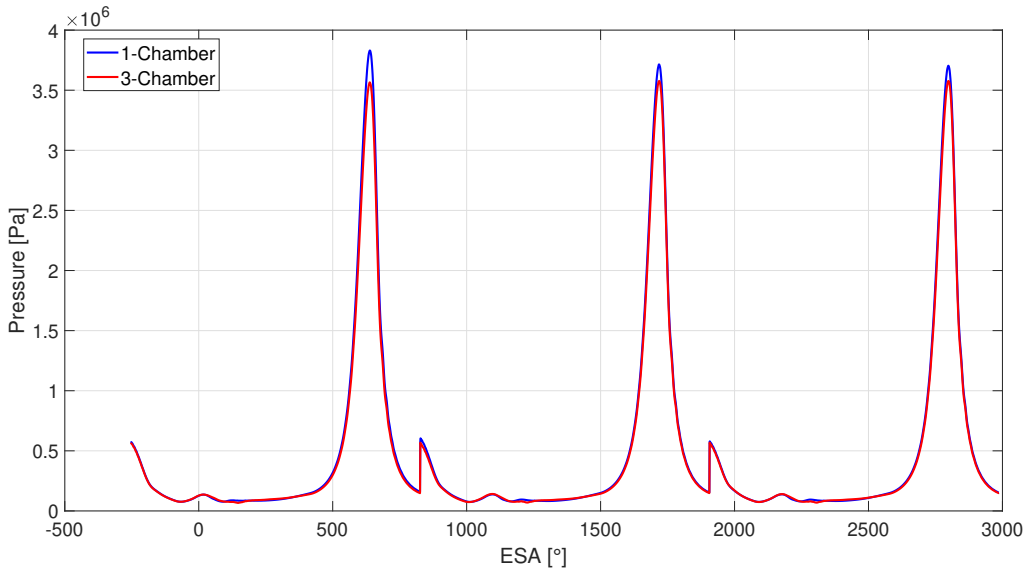
The figures below compare the results of mass and pressures inside the chamber 1 for the single and triple chamber configurations[4], highlighting a small difference in chambers filling due to the distinct gas dynamic processes between the simulations. It can be observed that the results are stable at the third cycle. The pressure discontinuities for 827 and 1907 are present because of the `setFields` initialization of the fields in order

[4]Chamber 2 and 3 plots are not reported here, but they show similar trends.

to simulate combustion. In fact, in case of combustion the peak pressure for the TDC position, which in this case can be considered related to a motored cycle, would be bigger, and the pressure plots would connect with the field values in those two points.



(a)



(b)

Figure 4.12: Volumetric efficiencies as function of ducts length.

It is evident that the single-chamber simulation achieves marginally higher pressure and greater volumetric efficiency, as the mass trapped per cycle is larger. However, by examining the values of the purity parameter, defined by equation (4.3), it can be observed

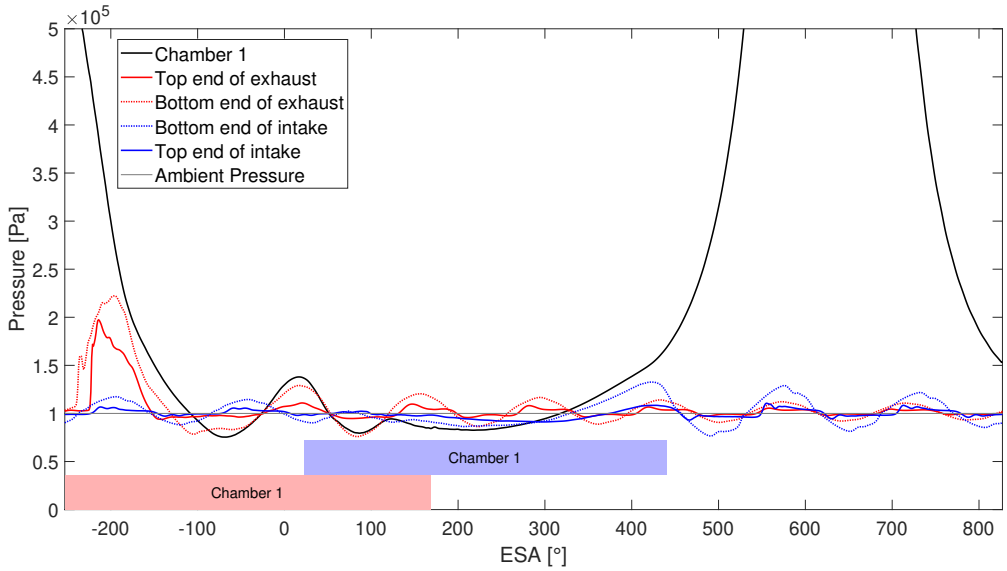
that this parameter is slightly higher for the three-chamber configuration. This suggests that, in the single-chamber configuration, there is a slightly larger fraction of residuals and recirculated combustion gases. Despite this, the trapped air mass in the single-chamber configuration is still greater.

$$\chi = \frac{m_{\text{tot}}}{m_{\text{air}}} = \frac{m_{\text{tot}}}{m_{\text{O2}}\left(1 + \frac{0.767}{0.233}\right)} \tag{4.3}$$

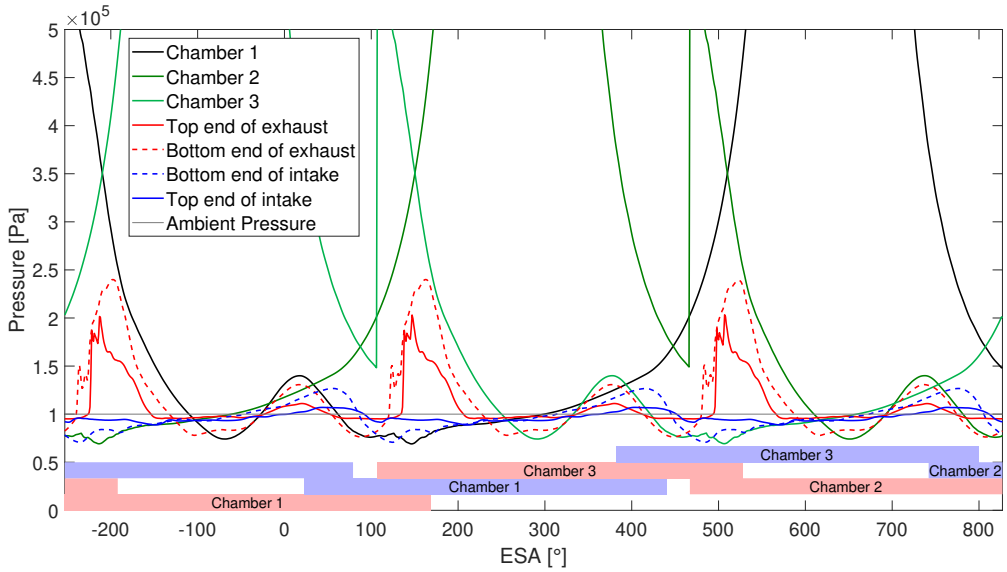
	1-chamber	**3-chamber**
λ_v	0.9832	0.96542
χ	0.92768	0.93022
Mass of air	0.26953 g	0.26466 g
Mass of other components	0.0210 g	0.0199 g
Total mass	0.29053 g	0.28456 g

Table 4.13: Comparison of volumetric efficiency and purity parameters for 1-chamber and 3-chambers configurations.

By examining the pressure plots in Figure 4.13, it can be inferred that the discrepancy between the two configurations arises from the distinct pressure trends they exhibit. In fact, the single-chamber configuration does not account for the interactions between two adjacent chambers when both face the exhaust and intake ports. This leads to varying inertial effects within the ducts, as simulating all three flanks of the rotor ensures that the intake and exhaust ducts are always in contact with the chambers. In contrast, this does not occur in the single-flank configuration.



(a) Pressure trends for the single chamber configuration.



(b) Pressure trends for the 3-chambers configuration.

Figure 4.13: The rectangles at the bottom of the graphs represents the intake phases of each chamber (in blue) and the exhaust phases (in red). Two pressure probes for each duct were set up: the dotted ones are near placed near the port (blue for intake, red for exhaust), while the solid line ones are placed to the top end of the ducts.

To further investigate the aforementioned statements and gain deeper insights into the phenomena occurring within the chambers, ParaView, a powerful open-source visualization and data analysis tool included in OpenFOAM, will be utilized. By employing ParaView, the simulation data can be analyzed, flow patterns visualized, and a better

understanding of the underlying mechanics contributing to the differences between the single-chamber and three-chamber configurations can be obtained. The differences resulting from simulating a single chamber instead of three are numerous. These differences can be observed for the cycle corresponding to eccentric shaft angles between 1907 and 2987 when the engine reaches a point where the subsequent cycles yield identical results, indicating that the wave motion in the ducts has stabilized. For a better understanding of the chamber's position, it is useful to consider these angles with reference to the first cycle, which has ESA values ranging from -253 to 827. This can be done simply subtracting 2160 (1080x2) from the ESA values between 1907 and 2987. This will provide a clearer perspective on the position of the chamber in relation to the opening of the ports.

The differences in pressure distribution between the single-chamber and three-chamber configurations can be analyzed starting from 2182 ESA, corresponding to 22 ESA of the first cycle. In the custom scale used for this study, pressure values are represented within a limited range of 0.8 bar to 2 bar. This scale was designed to focus on the range of interest; however, it is important to note that pressures in the chambers, as observed in previous plots, can exceed 6 bars. Consequently, values above 2 bar will not be distinguishable with this custom scale and this should be considered when interpreting the results.
For the three-chamber simulation, the exhaust duct exhibits a positive pressure, which can be observed in both the 4.14 and 4.13. This is not the case for the single-chamber configuration, where its intake duct has a negative pressure, lower than the ambient pressure.

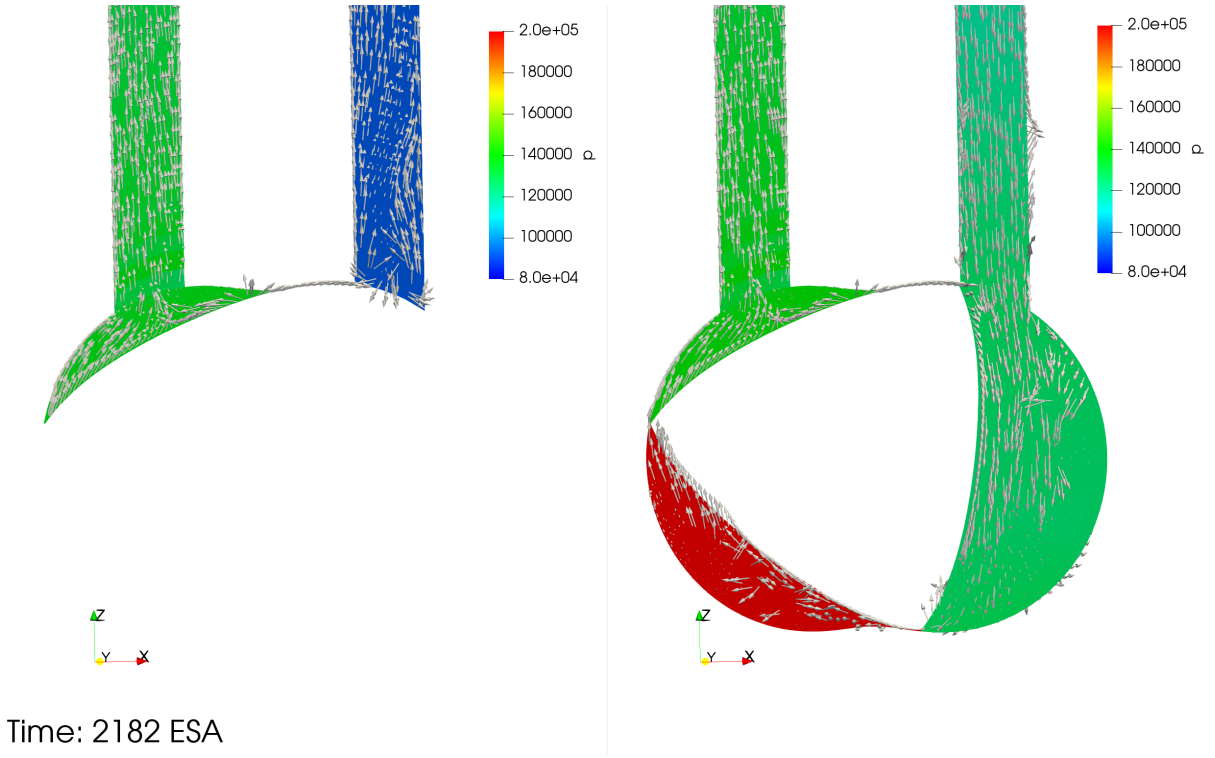


Figure 4.14: Comparison of pressure fields and velocity vectors directions at 2182 ESA (corresponding to 22 ESA).

This discrepancy is due to the fact that, for the single-chamber configuration, the intake duct has not been in contact with any chamber for an interval of about 670 ESA. This implies that the inertial motions and pressure oscillations within it are quite different from those of the three-chamber case, which was already drawing fresh charge into the preceding chamber (chamber 2) at the moment of chamber 1's IPO.

Similar to conventional engines, having a positive pressure at the intake valve opening can allow for better scavenging of the chamber, while having a negative pressure, as in the single-chamber case, is detrimental. Although the pressure of chamber 1 in both cases is greater than that within the intake and chamber 3 at the IPO, this pressure difference is much greater for the single-chamber case. As a result, a portion of the burnt gases recirculate inside the intake for this configuration, which becomes evident a few degrees past the IPO.

This is noticeable at 2212 ESA, where a greater fraction of CO2 (and consequently other combusted species) has flowed back into the intake duct. Furthermore, at this position, the backflow of these gases causes a more pronounced inversion of the velocity vectors in the first section of the intake.

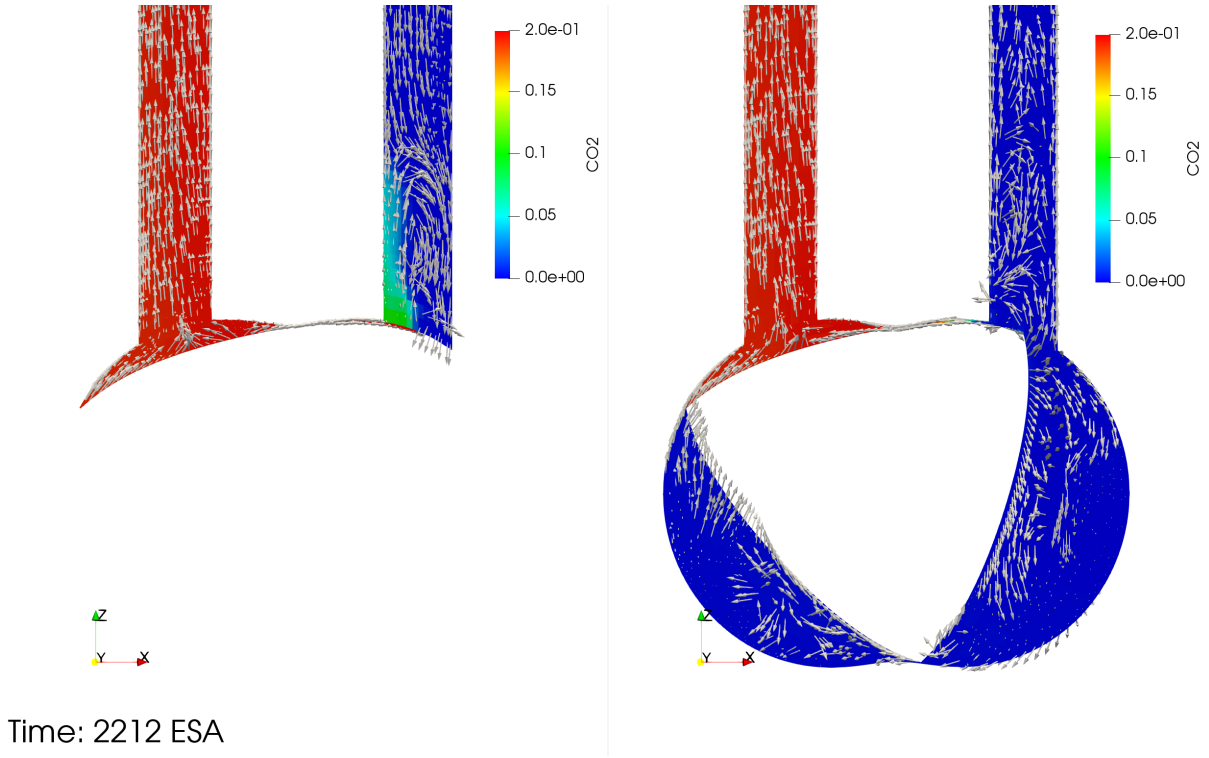


Figure 4.15: CO2 mass fraction comparison at 2212 (corresponding to 52 ESA).

At around 2192 ESA, the three-chamber case is in a situation where the pressures tends to equalize between the two chambers and the two ducts. Subsequently, in the final part of the burnt gas expulsion, the pressure in the exhaust duct decreases. However, due to the previously mentioned differences, the pressure in the intake will be quite different between the two configurations. In fact, at around 2206 ESA, it will still be negative for the single-chamber configuration, while it will be higher for the three-chamber case.

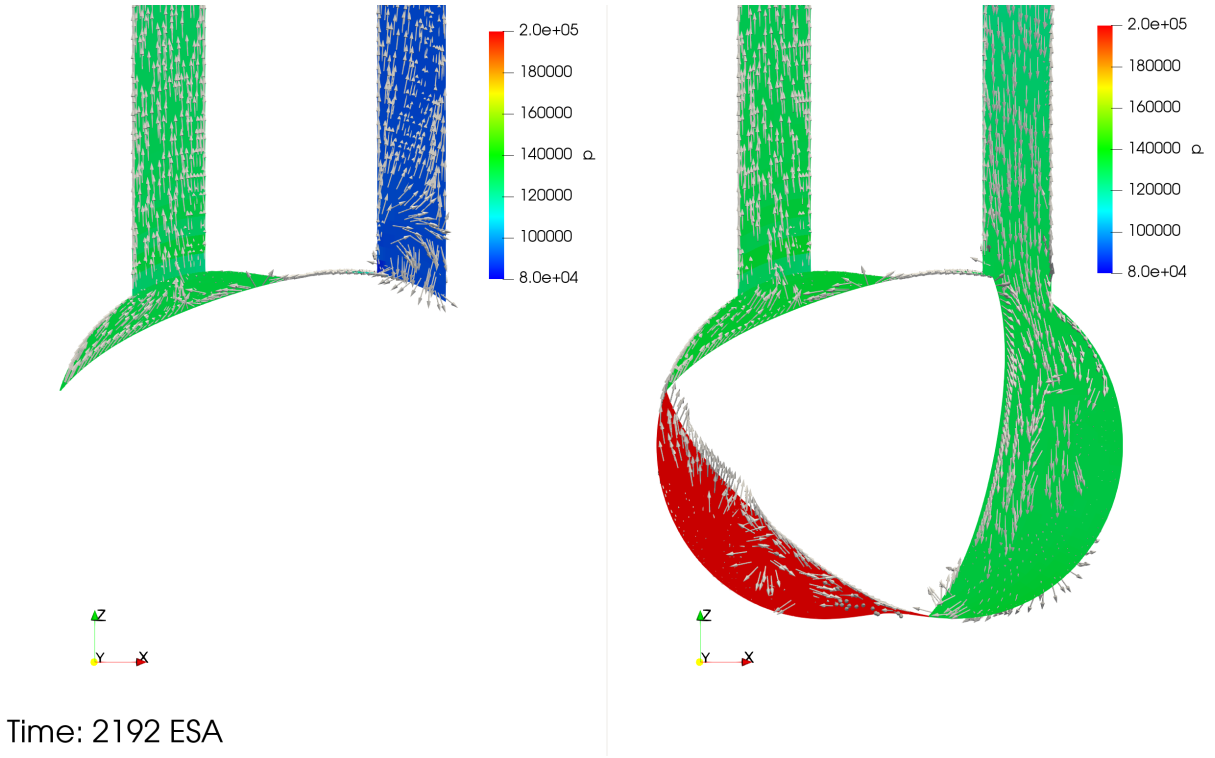


Figure 4.16: Comparison of pressure fields and velocity vectors directions at 2192 ESA (corresponding to 32 ESA).

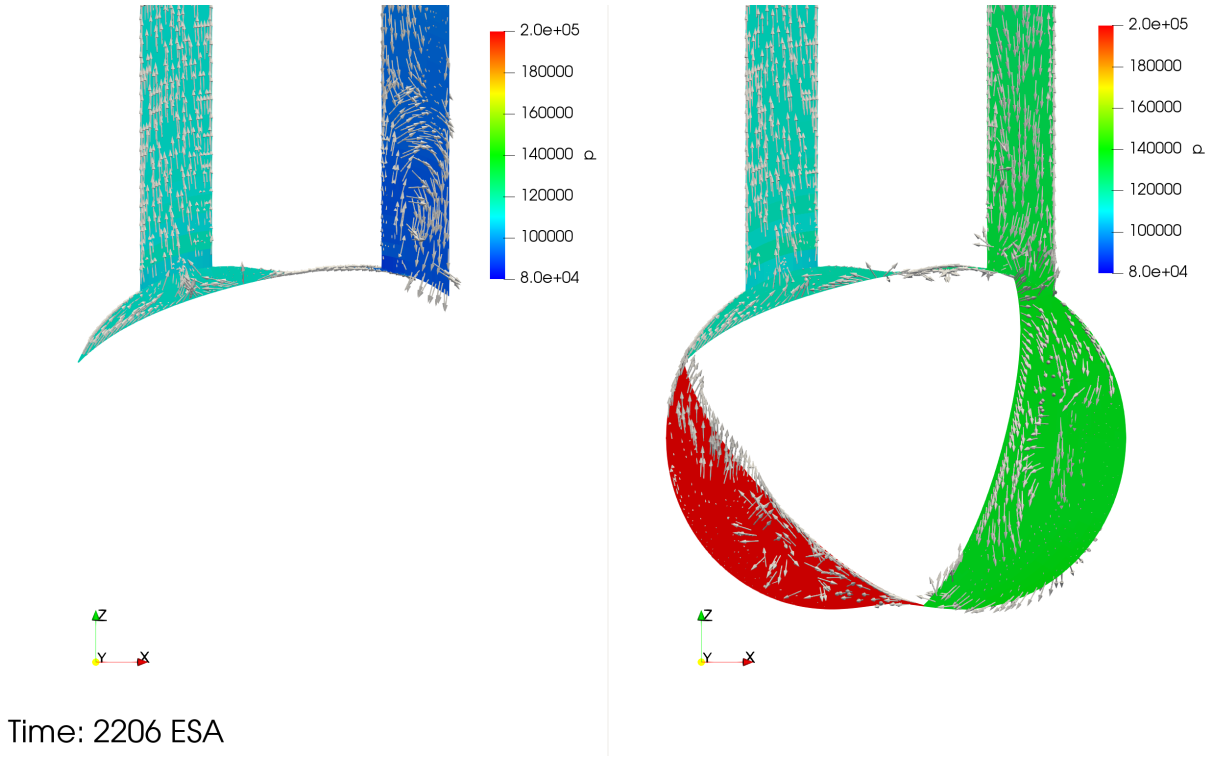


Figure 4.17: Comparison of pressure fields and velocity vectors directions at 2206 ESA (corresponding to 46 ESA).

This implies that, for the single-chamber case, the leakage of burnt gases will continue at this position, while for the three-chamber case, there will be a tendency for the fresh charge to enter chamber 1 towards the exhaust duct, this promoting chamber scavenging.

This is made evident by the figures 4.18 and 4.19, which show that O2 reaches the intake much earlier for the three-chamber case and in greater quantities. The O2 remains in the intake for a reduced range of ESA in the three-chamber case because it is then expelled or sent back to the intake by the overpressure caused by the chamber that begins the exhaust.This does not happen for the single-chamber case, as the lack of simulation for the two chambers causes the oxygen to remain in the exhaust duct for a longer time.

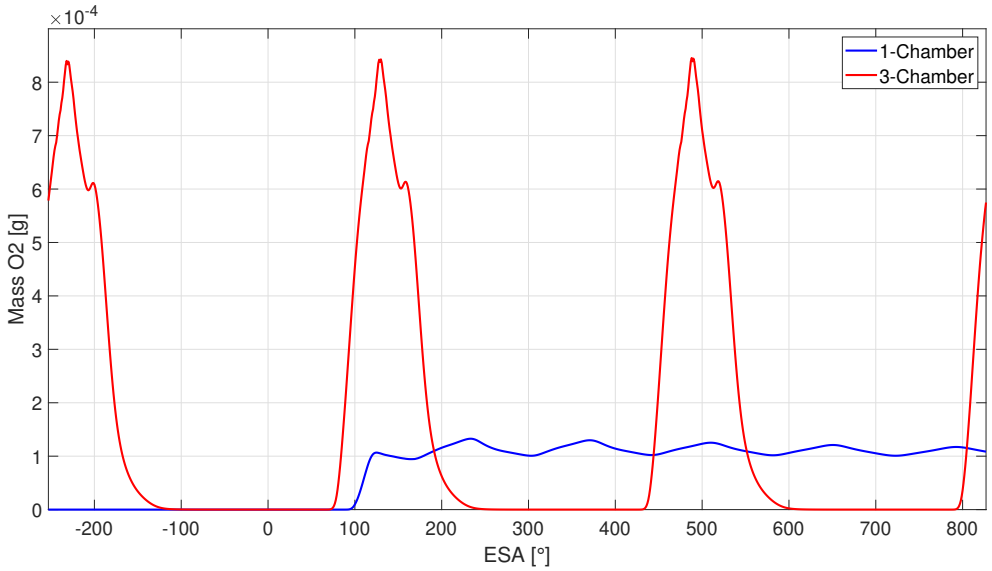


Figure 4.18: O2 mass comparison in the intake duct.

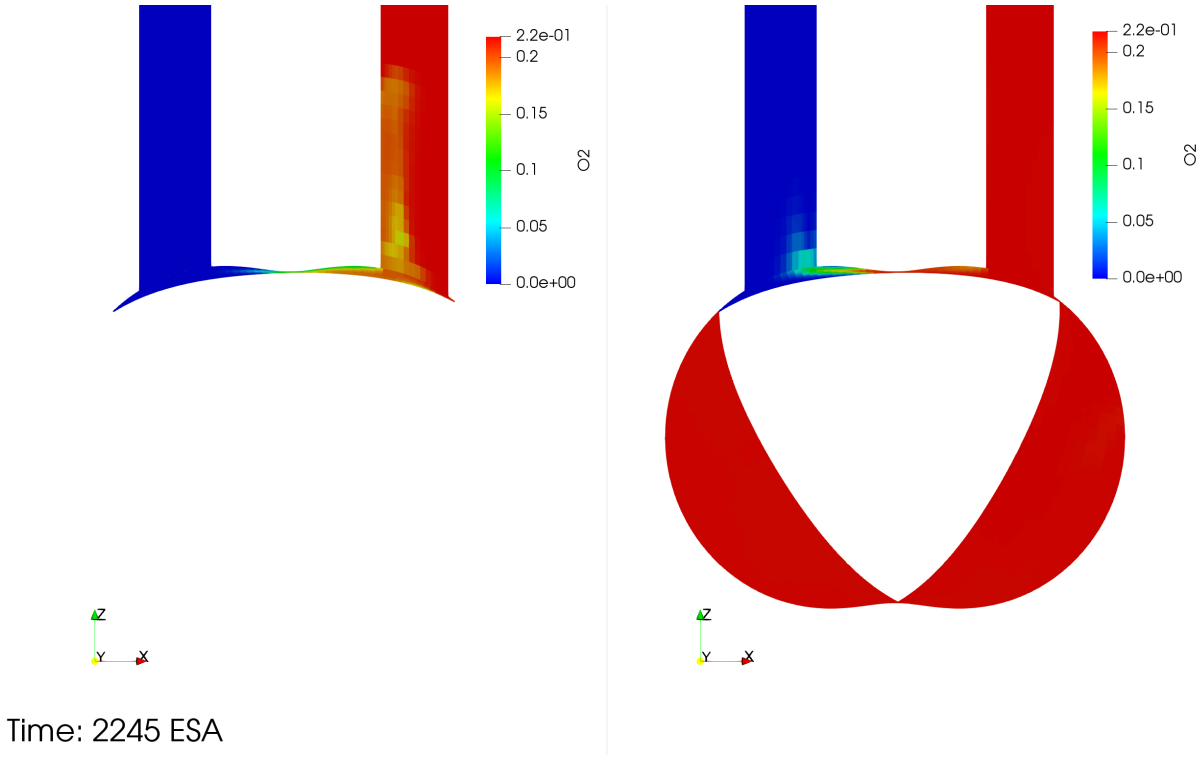


Figure 4.19: O2 mass comparison at 2245 ESA (corresponding to 85 ESA).

The effect of fresh charge leakage towards the exhaust would be positive (except for the unburned hydrocarbon emissions) if it were not for the fact that, probably due to the circular shape of the openings, part of the charge that recirculates in chamber 1 is coming

from the chamber that is finishing the intake phase (chamber 2), causing a lower filling of the latter (figure 4.20). This will, of course, also occur for chamber 1 when it is towards the end of its intake stroke, so to have a better visualization of the phenomenon, it is better to visualize this position, corresponding to 2578 ESA (figure 4.21).

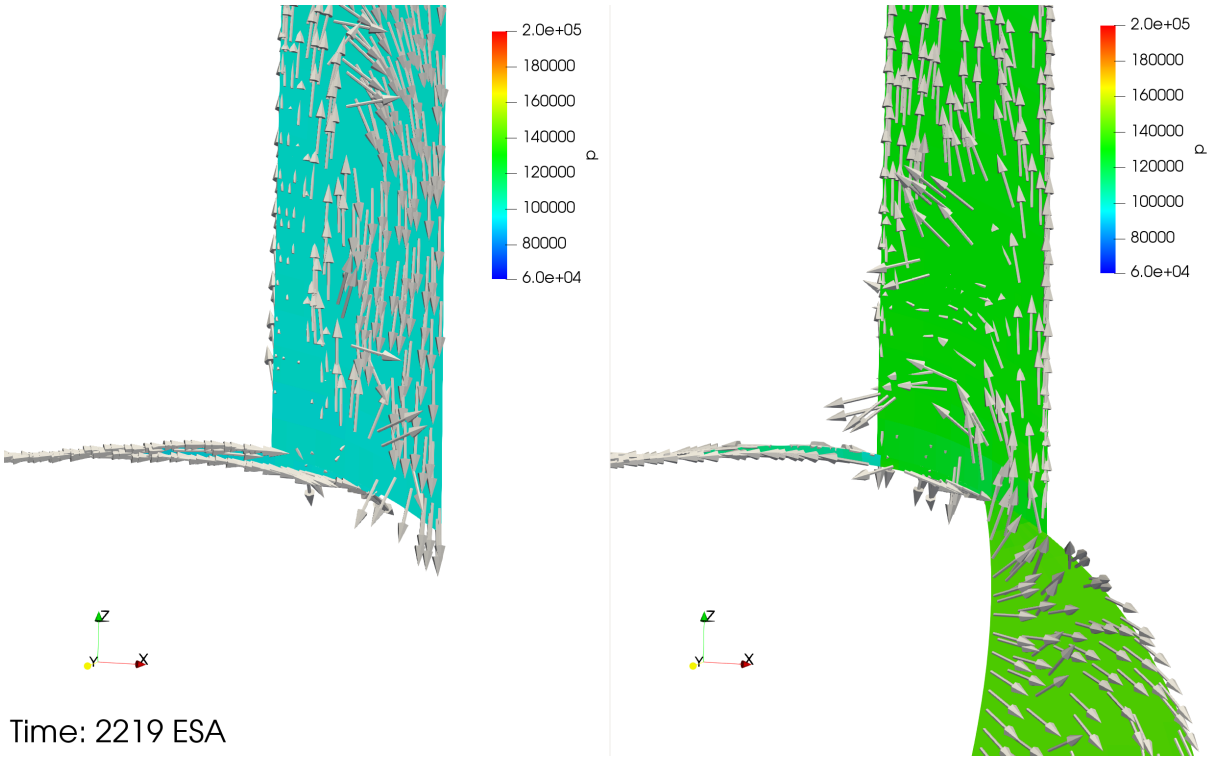


Figure 4.20: Comparison of pressure fields and velocity vectors directions at 2219 ESA (corresponding to 59 ESA).

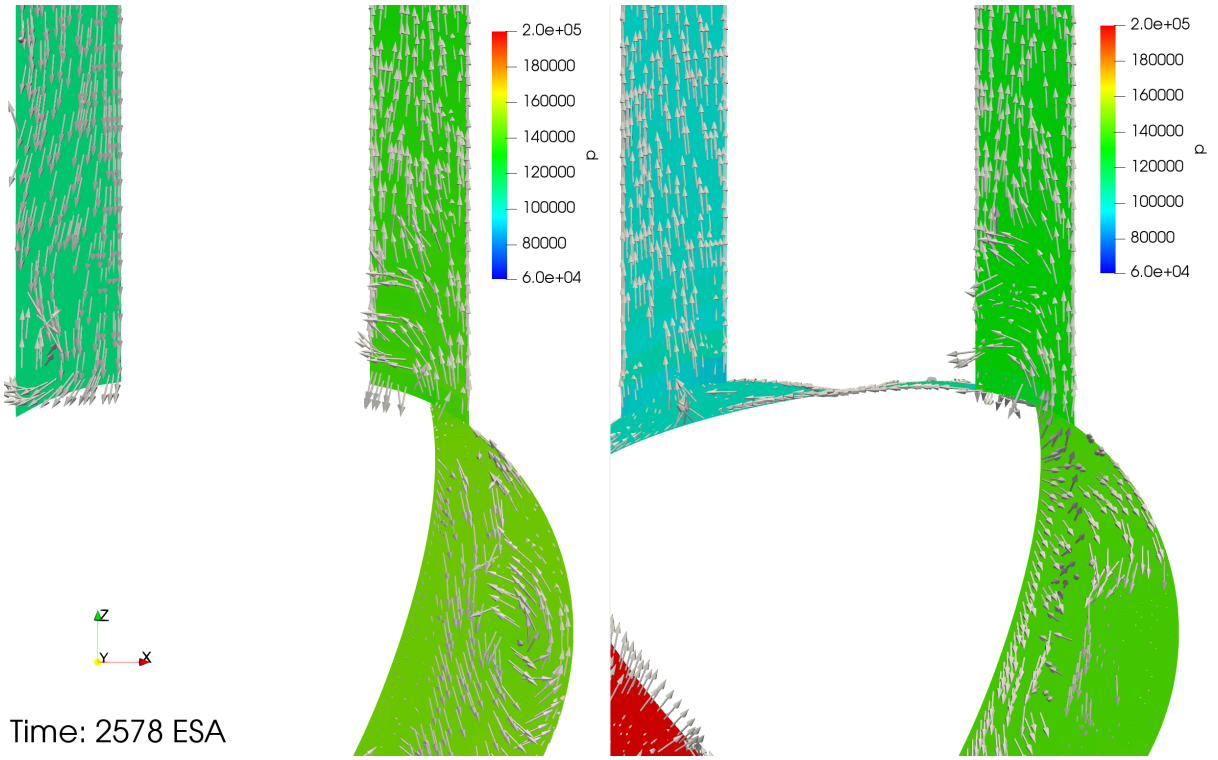


Figure 4.21: Comparison of pressure fields and velocity vectors directions at 2578 ESA (corresponding to 418 ESA).

As can be observed, chamber 1 is slightly over-pressurized compared to the intake, and this fact establishes a charge circulation from chamber 1 through the intake to chamber 3 (which is discharging).

This likely implies a lower volumetric efficiency and a lower trapped mass for the three-chamber configuration. However, it also explains why the three-chamber configuration has a higher degree of purity, as it has less passage of burnt gases towards the intake.

The lower trapped mass in the three-chamber configuration is also demonstrated by figure 4.22, where it can be noted that chamber 1 has a lower pressure compared to the other configuration.

Furthermore, at this position, in correspondence with the beginning of chamber 1's compression, it can be observed that the chamber has a higher pressure than the intake duct. Due to the sub-optimal shape of the ports, the release of this pressure causes high discharge velocities from the chamber towards the intake duct, resulting in a localized pressure reduction in its initial section. This phenomenon is accentuated in the three-chamber configuration, and is compounded by the previously mentioned charge motion towards the exhaust.

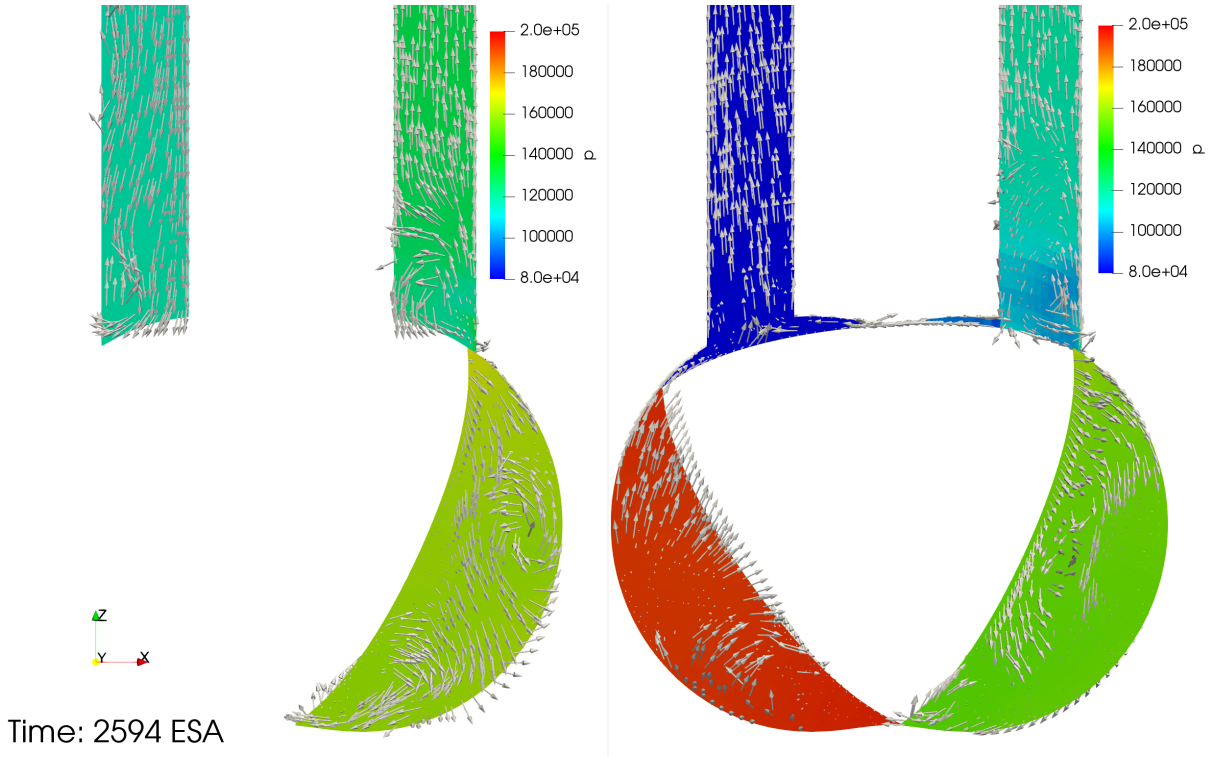


Figure 4.22: Comparison of pressure fields and velocity vectors directions at 2594 ESA (corresponding to 434 ESA).

The combination of these phenomena results in the typical counter-rotating vortices of the Wankel engine generated during the compression phase being less pronounced in the three-chamber configuration, as can be observed at 2599 ESA (figure 4.23). This probably means that there will be the generation of other out-of-plane flow features.

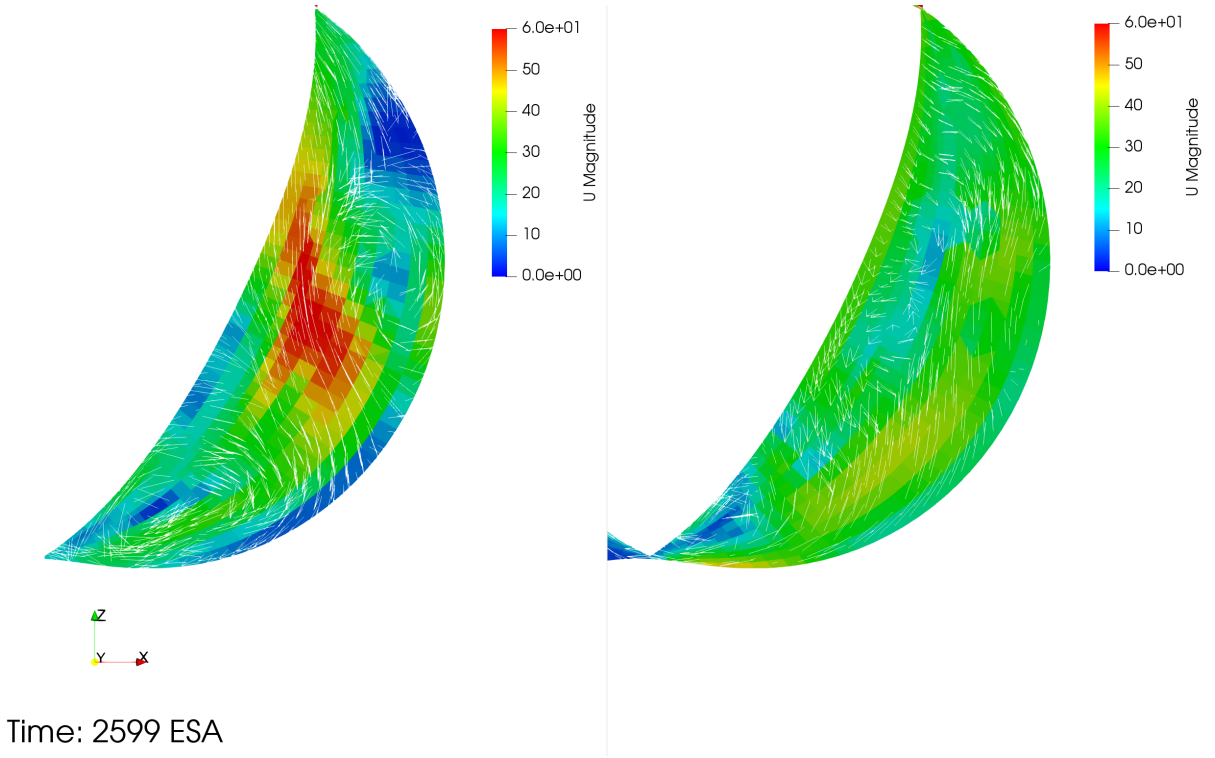
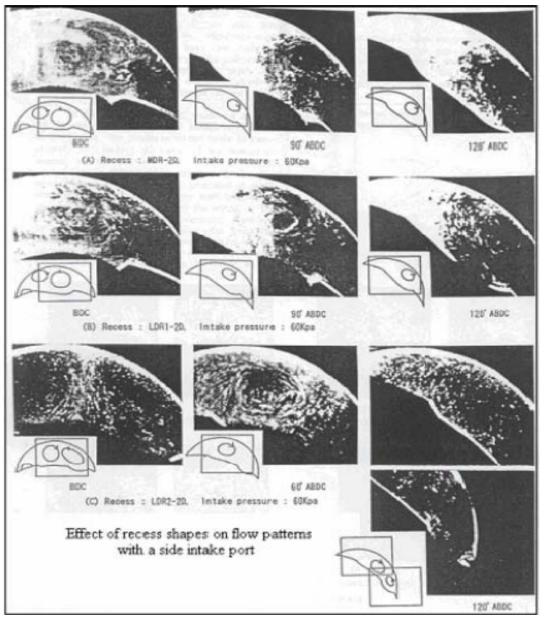
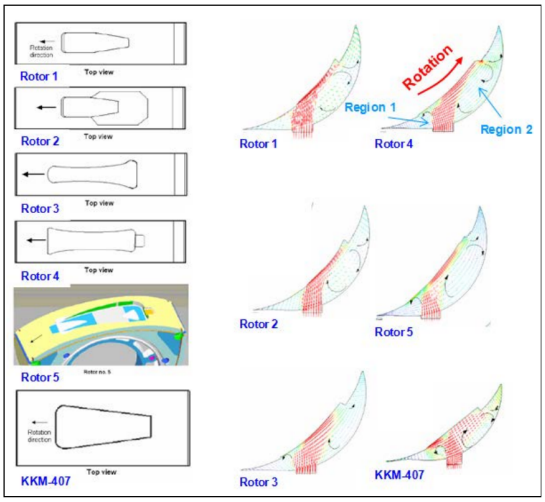


Figure 4.23: Comparison of pressure fields and velocity vectors directions at 2206 ESA (corresponding to 46 ESA).

This vorticity afore mentioned is consistent with findings in the literature, as numerous studies discovered this behaviour. For instance, studies by Hasegawa et al. [34] and Izweik et al. [35] have found this flow feature for all the studied configurations (Figure 4.24). In these researches it was observed that the flow field exhibits two regions for almost all the rotor recess configurations: one behind the inlet port with counterclockwise circulation and another in the leading flow part in the direction of rotation, displaying clockwise circulation at the rotor pocket (or recess) end. All the recess shapes demonstrated the same circulation effects, albeit with varying intensities. As observed earlier, vortex motions can be generated even in the absence of a rotor recess, at least for the single-chamber configuration. It is important to note that in the referenced studies, the rotor rotates counterclockwise, which is opposite to the clockwise rotation used in this simulation.



(a) Effect of rotor recess from [34].



(b) Effect of rotor recess from [35]

Figure 4.24: From these results, it can be easily seen that for all the configurations, similar vortical structures are formed.

In conclusion, it is evident that this ducts configuration does not promote optimal chamber filling. However, by examining, for instance, Figure 4.5, it can be observed that the duct length has an effect on the pressure inside the chamber, allowing for the anticipation or delay of the first pressure peak within the chamber. This suggests that testing additional configurations with varying duct lengths could lead to improved engine tuning and the avoidance of the previously explained detrimental phenomena. For example, adjusting the closure of the intake port to coincide with the point where fresh charge is still entering due to inertial effects, before any backflow occurs from the intake toward the exhaust.

5 | Conclusions

This work can be considered as a preliminary analysis concerning the Wankel peripherally ported rotary engine. The development of a methodology for generating the computational grid has been implemented by creating two new specific utilities, called `createWankelEngineFullMesh` and `createWankelEngineMesh`, respectively for creating the complete grid (3 chambers) and a single chamber (1 chamber). These were created by thoroughly analyzing the engine geometries and its kinematics, mainly referring to the work of Yamamoto[4]. The general procedure involved the creation of multiple meshes for different simulation timesteps and the interpolation of the various cells composing the grid for intermediate positions. A study on the effect of interpolation error on chamber volume depending on various setup parameters was conducted.

These discrete position meshes were created by rotating the stator points at each angular position and constructing the stator profile on top of it using appropriate equations. The methodology also includes the creation of the grid for the ducts (coupled via ACMI) and the possibility, through `snappyHexMesh`, to import the geometries of the recess and engine ducts on the simplified geometry using a `.stl` file. For this work, it was decided to use a small Wankel engine model, similar to the AIE 225 CS, originally designed as a range extender, but revisited for use as a propulsion system for UAVs.

Subsequent efforts were focused on achieving model convergence and solving issues related to ACMI coupling and non-conservation of mass and various quantities, especially along the stator surface. Once convergence was achieved and the model's consistency was verified, a preliminary gas-dynamic analysis of the engine was carried out, focusing on its filling and differences arising from simulating one chamber or three. Differences between these two configurations were observed in terms of final cylinder filling, motion, and flow features within the chamber.

Being a preliminary work, this comparison further validated the model's sensitivity to duct length and varying configurations. However, further studies, especially using the actual engine geometry and comparing with experimental data, are necessary for effective engine characterization and optimal tuning based on requirements. This work has led

to the development of a general baseline model for simulating Wankel engines of various sizes, which is reliable and highly versatile due to its functionalities.

However, due to the time-consuming full-cycle simulations and convergence issues encountered, combustion and spray modeling remained outside the scope of this research. This work provides a foundation for further studies focusing directly on combustion and injection types for this engine, initially using simpler models and later transitioning to more complex models incorporating chemical kinetics. Regarding injection, by developing an appropriate thermal model for the engine, it will be possible to model the formation of the wall film and fuel evaporation on the engine walls, and the effect of rotor recess shape, a critical aspect of the Wankel engine for to its geometry. This is particularly true if subsequent works focus on combustion of heavy fuels in small Wankel engines, which seems to be the trend in recent interests in the UAV sector.

Bibliography

[1] X Shen, AW Costall, M Turner, R Islam, A Ribnishki, JWG Turner, Giovanni Vorraro, Nathan Bailey, and Shaun Addy. Large-eddy simulation of a wankel rotary engine for range extender applications. In *Powertrain Systems for Net-Zero Transport*, pages 173–194. CRC Press, 2021.

[2] Donald Kennedy and Simon P Philbin. Techno-economic analysis of the adoption of electric vehicles. *Frontiers of Engineering Management*, 2019.

[3] Bastian Beyfuss, Lukas Flicker, Thomas Gotthard, Peter Hofmann, Felix Zahradnik, Christian Krenn, and Georg Lubich. Evaluation of spark-ignited kerosene operation in a wankel rotary engine. Technical report, SAE Technical Paper, 2021.

[4] Kenichi Yamamoto. *Rotary engine*. Sankaido, 1981.

[5] Giovanni Vorraro, Matthew Turner, and James WG Turner. Testing of a modern wankel rotary engine-part i: Experimental plan, development of the software tools and measurement systems. Technical report, SAE Technical Paper, 2019.

[6] J. Lion. Wankel rotary engine from mazda rx-7. Wikimedia Commons, September 2005. URL `https://commons.wikimedia.org/wiki/File:Wankel_Rotary_Engine_from_Mazda_RX-7.jpg`. Accessed: February 20, 2023.

[7] Michael Peden, Matthew Turner, James WG Turner, and Nathan Bailey. Comparison of 1-d modelling approaches for wankel engine performance simulation and initial study of the direct injection limitations. 2018.

[8] Chol-Bum M Kweon. A review of heavy-fueled rotary engine combustion technologies. 2011.

[9] James Turner, Matthew Turner, Giovanni Vorraro, and Toby Thomas. Initial investigations into the benefits and challenges of eliminating port overlap in wankel rotary engines. *SAE International Journal of Advances and Current Practices in Mobility*, 2020.

[10] James Turner, Matthew Turner, Reza Islam, Xuankun Shen, and Aaron Costall.

Further investigations into the benefits and challenges of eliminating port overlap in wankel rotary engines. *SAE International Journal of Advances and Current Practices in Mobility*, 2021.

[11] N Bailey and L Louthan. The compact sparcs wankel rotary engine. *JSAE Journal*, 2015.

[12] T. J. Norman. A performance model of a spark ignition wankel engine: Including the effects of crevice volume, gas leakages, and heat transfer. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1983.

[13] Giovanni Vorraro and James Turner. Testing of a modern wankel rotary engine-part iv: overall mechanical and thermal balance. Technical report, SAE Technical Paper, 2022.

[14] A. Nagao, H. Ohzeki, and Y. Niura. Present status and future view of rotary engines. In *Proceedings of the International Symposium on Alternative and Advanced Automotive Engines*, pages 183–201, Vancouver, British Columbia, Canada, 1987.

[15] Ralf Pfeifer. Drehkolbenmotor DKM 54. Wikimedia Commons, 2005. URL `https://commons.wikimedia.org/wiki/File:DrehkolbenmotorDKM54.JPG`. License: GNU FDL.

[16] Susan Chira. A new test for mazda's head. *The New York Times*, Jan 1985. URL `https://www.nytimes.com/1985/01/12/business/a-new-test-for-mazda-s-head.html`.

[17] Masaki Ohkubo, Seiji Tashima, Ritsuharu Shimizu, Suguru Fuse, and Hiroshi Ebino. Developed technologies of the new rotary engine (renesis). Technical report, SAE Technical Paper, 2004.

[18] Ritsuharu Shimizu, Haruo Okimoto, Seijo Tashima, and Suguru Fuse. The characteristics of fuel consumption and exhaust emissions of the side exhaust port rotary engine. *SAE transactions*, 1995.

[19] 160SX (photographer). Mazda r26b. Wikimedia Commons, June 2007. URL `https://commons.wikimedia.org/wiki/File:MAZDA_R26B_01.jpg`. Accessed: February 21, 2023.

[20] Advanced Propulsion Centre. Light duty vehicle <3.5t roadmap 2020. `https://www.apcuk.co.uk/app/uploads/2021/09/https___www.apcuk_.co_.uk_app_uploads_2021_02_Exec-summary-Product-Roadmap-LDV-final.pdf`, 2021. Accessed on: March 6, 2023.

[21] International Council on Clean Transportation. European vehicle market statistics pocketbook 2021/2022. https://theicct.org/wp-content/uploads/2021/12/ICCT-EU-Pocketbook-2021-Web-Dec21.pdf, 2021. Accessed on: March 6, 2023.

[22] International Council on Clean Transportation. European vehicle market statistics pocketbook 2020/2021. https://theicct.org/sites/default/files/publications/ICCT_EU_Pocketbook_2020_Web_Dec2020.pdf, 2021. Accessed on: March 6, 2023.

[23] Advanced Innovative Engineering (UK) Ltd.). Wankel rotary engine 225cs 40bhp. Product Datasheet, 2021. URL https://www.aieuk.com/225cs-40bhp-wankel-rotary-engine. Last accessed on: March 1, 2023.

[24] J W Walker and R E Mount. The stratified charge rotary engine. *John Deere Review*, 1986.

[25] C Jones. An update of applicable automotive engine rotary stratified charge developments. In *SAE Technical Paper*. SAE International, 1982.

[26] F Feller. The 2-stage rotary engine—a new concept in diesel power. *Proceedings of the Institution of Mechanical Engineers*, 1970.

[27] A.I.E.U. Ltd. 225cs jp8 fuel test. https://www.aieuk.com/video/, 2014. Accessed February 20 , 2023.

[28] Hrvoje Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. 1996.

[29] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[30] Pijush K Kundu, Ira M Cohen, and David R Dowling. *Fluid mechanics*. Academic press, 2015.

[31] Joel H Ferziger, Milovan Perić, and Robert L Street. *Computational methods for fluid dynamics*, volume 3. Springer, 2002.

[32] Openfoam user guide: Mesh generation with the blockmesh utility, 2021. URL https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.3-mesh-generation-with-the-blockmesh-utility. Accessed: 2023-03-14.

[33] Giancarlo Ferrari, Angelo Onorati, and Gianluca D'Errico. *Internal combustion engines*. Società Editrice Esculapio, 2022.

[34] Yasuaki Hasegawa and Kouichi Yamaguchi. An experimental investigation on air-fuel mixture formation inside a low-pressure direct injection stratified charge rotary engine. Technical report, SAE Technical Paper, 1993.

[35] Husni Taher Izweik. *CFD investigations of mixture formation, flow and combustion for multi-fuel rotary engine.* PhD thesis, BTU Cottbus-Senftenberg, 2010.

List of Figures

1.1 Comparison of components number with conventional engine 3
1.2 Section and main parts of the engine. 5
1.3 Engine strokes 6
1.4 Strokes comparison with conventional piston engines. 7
1.5 Rotor 8
1.6 Rotor housing 8
1.7 Shaft. 9
1.8 Sealings 10
1.9 Size and specific power comparison. 11
1.10 SPARCS system. 13
1.11 Flame propagation model 14
1.12 Wankel engine pressure curve 14
1.13 Old concepts 15
1.14 NSU engines 16
1.15 Kenichi Yamamoto, his team and the Cosmo Sports Cars. 17
1.16 Ro 80 (NSU) 17
1.17 Spider (NSU) 18
1.18 Mazda Engines 19

2.1 Representation of a general polyhedron cell 33
2.2 Upwind interpolation scheme. 36
2.3 Linear interpolation scheme. 36
2.4 Linear upwind interpolation scheme. 37
2.5 Skewness representation. 39
2.6 Non orthogonality vectors. 40
2.7 Matrix assembly. 44
2.8 Simplified cell schematization. 45
2.9 Turbulence scales. 47

3.1 Trochoid geometries 56

3.2 Stator and rotor curves with MATLAB. 57
3.3 Geometrical generation of the stator curve 57
3.4 Stator equations comparison with MATLAB. 59
3.5 Rotor equations comparison with MATLAB. 61
3.6 Oscillation angle. 62
3.7 Compression ratio. 64
3.8 1-chamber and 3-chambers meshes 65
3.9 Polylines and mesh of the single chamber at $\theta = 0$. 68
3.10 2D polylines plot for the 3-chambers configuration at $theta = 0$. 69
3.11 Polylines for the 3-chambers configuration. 70
3.12 Mesh resolution parameters. 72
3.13 Blocks of ducts mesh. 73
3.14 `blockMesh` block. 74
3.15 Deformation of stator boundary cells. 79
3.16 Dimensions of AIE225 CS. 80
3.17 Power and torque curve of AIE225 CS. 80
3.18 Engine displacement law. 82
3.19 Engine displacement law error. 83
3.20 Interpolation error. 84

4.1 Mesh non-orthogonality and aspect ratio for the full cycle. 88
4.2 Patches of the simulation. 89
4.3 Courant number for 0.125 s time step. 105
4.4 Trapped mass plot for 5 cycles. 106
4.5 Effect of exhaust length on chamber pressure. 107
4.6 Effect of intake length on chamber pressure. 108
4.7 Effect of exhaust length on trapped mass. 108
4.8 Effect of intake length on trapped mass. 109
4.9 Effect of intake length on chamber temperature. 109
4.10 Volumetric efficiencies as function of ducts length. 111
4.11 Courant number for 0.0125 s time step. 112
4.12 Volumetric efficiencies as function of ducts length. 113
4.13 Pressure trends comparison for 1-chamber and 3-chambers configuration. . 115
4.14 Comparison at 2182 ESA: p and velocity vectors. 117
4.15 Comparison at 2212 ESA: CO2 mass. 118
4.16 Comparison at 2192 ESA: p and velocity vectors. 119
4.17 Comparison at 2206 ESA: p and velocity vectors. 120

4.18 Comparison of O2 mass recirculation. 121
4.19 Comparison at 2245 ESA: O2 mass. 121
4.20 Comparison at 2219 ESA: p and velocity vectors. 122
4.21 Comparison at 2578 ESA: p and velocity vectors. 123
4.22 Comparison at 2594 ESA: p and velocity vectors. 124
4.23 Comparison at 2599 ESA: U magnitude and velocity vectors. 125
4.24 Examples of rotor recess effects 126

List of Tables

3.1 Main Wankel dimensions 56
3.2 Main AIE225 CS specs. 81

4.1 Mesh parameters and Wankel geometry 88
4.2 CheckMesh summary 89
4.3 Boundary conditions for p, U, and T 90
4.4 Field values for different cell zones 93
4.5 Boundary conditions for chemical species 94
4.6 Boundary conditions for default mass fraction of additional species, mixture fraction, and mixture fraction variation. 95
4.7 Initial conditions for the thermal diffusivity. 96
4.8 Initial conditions for the turbulent kinematic viscosity 97
4.9 Initial conditions for the turbulent kinetic energy 97
4.10 Initial conditions for the rate of dissipation of turbulent kinetic energy. . . 98
4.11 Pressures [bar] for all the configurations 106
4.12 Ports timing for different chambers. 112
4.13 Comparison of volumetric efficiency and purity parameters for 1-chamber and 3-chambers configurations. 114

List of Acronyms

RE rotary engine

CR compression ratio

ICE internal combustion engines

IPO intake port opening

EPO exhaust port opening

IPC intake port closing

EPC exhaust port closing

CAD crank angle dregree

ESA eccentric shaft angle

BDC bottom dead center

TDC top dead center

HC hydrocarbon

rpm revolutions per minute

SPARCS self-pressurizing air-cooled rotor system

ACMI Arbitrary Coupled Mesh Interface

AIE advanced innovative engineering

CI compression ignition

SCRE stratified charge rotary engine

BEV battery elctric vehicles

FCEV fuel cell electric vehicles

HEV hybrid electric vehicles

REEV range-extended electric vehicles

EV electric vehicles

BSFC brake specific fuel consumption

DI direct injection

SI spark ignition

UAV unmanned aerial vehicle

AvGas aviation gasoline

BMEP brake mean effective pressure

SIP sheet metal insert process

CFD computational fluid dynamics

FV finite volume

CV control volume

List of Symbols

Operator	Description
∇	nabla operator (gradient)
$\cdot$	dot product or scalar product
Δ	Laplacian operator
δ_{ij}	Kronecker delta
$\int$	integral operator
∂	partial derivative
tr	trace operator
$\overline{\overline{I}}$	identity matrix

Subscript or Superscript	Description
$_i$	Index representing the i-th direction or component
$_j$	Index representing the j-th direction or component
$_P$	Referring to cell centroid P
$_N$	Referring to neighbouring cell centroid N
$_f$	Referring to cell face f
$_n$	normal component
^{n}	new time level
^{o}	old time level
$'$	term related to turbulence
^{T}	Transposed

Acknowledgements

I would like to express my profound appreciation and gratitude to my advisor, Prof. Lucchini Tommaso, for his invaluable guidance and unwavering support throughout this thesis project. His expertise, enthusiasm, and dedication to his work have been truly inspiring and have significantly contributed to my academic growth.

I am deeply grateful to my family for their constant love, encouragement, and support in every aspect of my life. Their belief in my abilities and their unwavering confidence in my success have been instrumental in shaping my academic journey.

I wish to dedicate the efforts of the past months to my mother Graziana and my grandfather "Dino," who have always held a special place in my heart. Their memory continues to inspire and motivate me in my pursuits.

Lastly, I would like to extend my sincere gratitude to my uncle Giampaolo for his invaluable assistance and steadfast presence throughout my university career. His guidance and encouragement have been vital to my academic achievements, and I am truly grateful for his unwavering support.