POLITECNICO DI TORINO

Corso di Laurea Magistrale in
Ingegneria Energetica e Nucleare

Tesi di laurea magistrale

Detached eddy simulation of an aircraft jet flow using an open-source CFD code

Relatori
Prof. Marco Carlo Masoero (Politecnico di Torino)
Prof. Roberto Paoli (University of Illinois at Chicago)

Candidato
Riccardo ROMA
Matricola: 257517

Anno accademico 2019/2020

Abstract

This thesis is a preliminary work of the project "High-performance computing and data-driven modeling of aircraft contrails" granted by the US NFS (National Science Foundation) and with PI Prof. Roberto Paoli of the Illinois University of Chicago (UIC). In the work is analyzed the *Jet Phase* of the contrail formation without considering soot particles exiting from the aircraft engine; whose drain nozzle geometry has been considered similar to the NASA acoustic research nozzle ARN-2. The fluid dynamic study is performed using only open source software, in particular Gmsh for the mesh generation and OpenFOAM as CFD code. Different geometries and turbulence models are explored, with a great attention on the results obtained with the *Detached Eddy Simulations* based on the $k-\omega$ SST model. Furthermore, in the thesis are described in detail all the settings to properly set up the OpenFOAM's solver *rhoPimpleFoam* for jet flows at sonic conditions, therefore the second aim of the work is to give to the user a comprehensive guide to launch an OpenFOAM simulation for this kind of flows.

Keywords:OpenFOAM, Gmsh, Detached Eddy Simulations, Sonic Jets, ARN-2 nozzle

Acknowledgments

First, I would like to express my gratitude to my academic advisors, Professor Roberto Paoli, for his advice and suggestions and Professor Marco C. Masoero, for his kindness and helpfulness despite the distance. I would also like to thank Mrs. Jenna Stephens for her patience and availability during my stay in Chicago and to be always ready to cheer-up me and the other students of the TOP-UIC project during the pandemic. A thanks goes also to Professor's Paoli assistant Sibo Li, who helped me with OpenFOAM and was always available for questions and clarifications. During the time spent in Chicago, I have been surrounded by friends that shared with me this incredible experience and made me feel less the distance from my country. For this I want to thank all the Italian students who left with me the Politecnico to live this amazing experience. A thanks goes also to my classmates back in Turin, Edoardo and Francesco, who unburdened me from university pains and have given me wonderful memories that I will always bring within. Every time I come back home, in Puglia, I have friends who are always ready to welcome me with open arms and make me fell like I never went away, for this I want to thank Andrea, Giuseppe and Cosimo. Last but not the least the biggest and priceless thanks goes to my family, they allowed me to study away from home, first in Turin and then in Chicago. They supported me in every decision and they gave me the strength to complete these five years of university. They have been the lighthouse in these last dark months, when the Coronavirus pandemic and the miles away from home would have lower anyone's mood. Thank to my mother, for her blind love and for always making believe me in myself, thank to my father who thought me how to take life lightly and the values honesty and kindness, thank to my sister who is always behind me with her hidden and silent gestures, if today I am what I am is mainly thanks to them.

List of abbreviations

- **ARN-2** Acoustic Research Nozzle 2
- **CFD** Computational Fluid Dynamics
- **CV** Control Volume
- **DES** Detached Eddy Simulation
- **DDES** Delayed Detached Eddy Simulation
- **GAMG** Generalized geometric-algebraic multi grid
- **LES** Large eddy simulation
- **NFS** National Science Foundation
- **OpenFOAM** Operation Field and Manipulation
- **PbiCG** Preconditioned bi-conjugate gradient
- **PbiCGStab** Stabilized preconditioned bi-conjugate gradient
- **PI** Principal investigator
- **PISO** Pressure Implicit with splitting of operators
- **RANS** Reynolds-Average Navier Stokes
- **SGS** Sub-grid scales
- **SIMPLE** Semi-Implicit method for Pressure Linked Equations
- **SST** Shear Stress transport
- **UIC** University of Illinois at Chicago

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis work is part of a preliminary study of the project "High-performance computing and data-driven modeling of aircraft contrails", granted by the US NFS (National Science Foundation) and with PI Prof. Roberto Paoli of the Illinois University of Chicago (UIC). The aim of the project is the prediction of contrail formation in fully-three dimensional turbulent jets exhausting the aircraft nozzle exit using high fidelity LES simulations and use the data obtained by these simulations to train Artificial Neural Networks (ANN) in order to have an accurate model of the contrail structure. The main target of this thesis work is to analyze the first phase of the contrail formation, the so called *Jet Phase*, where the aircraft jet is expanded into the atmosphere, mixed with the ambient air and cooled down to the atmospheric temperature. This phase accours in the first 30 m behind the jet engine and it is assumed that the jet expansion is not influenced by the aircraft vortex formation. As a preliminary work, the simulations performed in the thesis take into account only the exhausted gases of the engine without adding soot particles that can act as nucleation sites for for the sublimation of the atmospheric water vapor. The simulations are run on OpenFOAM, an open source CFD code, and they are performed in an increasing level of turbulence model and geometry complexity. First are run simulations using the $k-\omega$ SST RANS model on an axisymmetric grid with periodic rotating boundary conditions, then it is considered a full 3D geometry on which is applied the same RANS model and finally the $k-\omega$ SST DES model which is an hybrid model between RANS and LES. On all the three cases the flow is simulated both considering and not considering the duct surrounding the drain nozzle of a CFM-56 engine, whose geometry has been considered similar to the one of the ARN-2 acoustic reference nozzle used at the NASA Glenn Research Center. In particular the ARN-2 nozzle geometry has been rescaled to account the exit diameter of 0.610 m of the CFM-56 engine. The set-up of the OpenFOAM solver to perform the simulations, is made on the data for the Axisymmetric near-sonic jet validation case of the NASA Langley Research, these flow data are for an unheated jet exiting at $Ma = 0.985$ from the ARN-2 nozzle and can be easily downloaded at [1]. In the work one of the main challenges has been the creation of a suitable structured grid to run the simulation with the $k-\omega$ SST DES turbulence model. For this model it is essential a good grid refinement in order to caught as much as possible turbulent length scales using the LES part of the model, without having an excessive computational cost. For this task it has been tried to generate a mesh with a grid size similar to the works of [2] ,[3] and [4] where it is simulated a fully

turbulent jet near sonic condition using a pure LES approach.
To account the large time consuming of all the performed simulations, the capability of OpenFOAM to run simulation simulation in parallel on different computer cores has been used. In particular the axisymmetric and the less refined 3D simulations have been run on Dragon the UIC cluster, while the most refined 3D cases have been run at the supercomputing infrastructure "Theta" (Cray XC40 with second generation Intel Xenon Phi processor) at the Argonne National Laboratory. The work is organized according to the following pattern. In chapter 2 are presented the governing equation of fluid dynamics together with the mathematical model to describe the turbulence phenomenon. In Chapter 3 is given a brief introduction to the OpenFOAM software and the finite volume method to discretize the Navier-Stokes equation is explained. Chapter 4 gives an overview of the PIMPLE algorithm used in compressible flow solvers and the parameters for the simulation set-up are tested on the NASA near sonic validation case. In chapter 5 are reported the boundary and initial conditions together with the results for the axisymmetric simulations, while in chapter 6 after the presentation of Gmsh (the open source software used to generate the 3D structured meshes), are reported the results for the 3D simulation with the $k-\omega$ SST and $k-\omega$ SST DES turbulence models. Finally, the last chapter is dedicated to the conclusion and the further developments of the work.

Chapter 2

Governing equations and modeling

In this chapter are presented the governing equation of fluids' motion and it is briefly explained the concept of *turbulence*. The main model techniques to model the unsteady and chaotic behavior of the fluid's physical quantities when turbulence arise are explained and finally, these techniques are compared with their pros and cons. As a matter of simplicity, in this thesis work, all the equations will be written according to the Einstein's tensor notation.

2.1 Governing Equations

The governing equations for fluids are the continuity equation (2.1), the momentum equation (2.2) and the energy balance equation (2.3).

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_j)}{\partial x_j} = 0 \tag{2.1}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) + \rho f_i \tag{2.2}$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial (\rho E u_j)}{\partial x_j} = -\frac{\partial (p u_j)}{\partial x_j} + \frac{\partial (u_i \sigma_{ij})}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + S_E \tag{2.3}$$

In the above equations, f_i is any force applied to the fluid, S_E is an energy source term and the tensor $\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}$ which also appears in (2.2) represents the viscous stresses, that in a Newtonian fluid are proportional to the rates of deformation. In (2.3) q_j is the heat flux in the j-direction (if there are no heat sources it can easily computed using the Fourier law $q_j = -k\frac{\partial T}{\partial x_j}$), while $E = e + \frac{1}{2}(u^2 + v^2 + w^2)$ is the sum of internal (thermal) energy e and kinetic energy $\frac{1}{2}(u^2 + v^2 + w^2)$. Normally, this term includes the gravitational potential energy, but it possible to regard the gravitational force as a body force, which does work on the fluid element as it moves through the gravity field. To close the system a further equation to relate p and e to the variables ρ and T is required. For compressible flows, this can be easily achieved using the well known equations of state for an ideal gas,

$$p = \rho R T \text{ and } e = c_v T. \tag{2.4}$$

The complete derivation of the equations can be found in [5].

2.1.1 Incompressibility

At low Mach numbers, the density of the fluid can be considered constant. This reduces (2.1) to:

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{2.5}$$

This allows to simplify the governing equation since there is no need to couple the energy equation (2.3) with the momentum equation (2.2) that now is only coupled with the simplified continuity equation (2.5). The new momentum equation can be written as:

$$\frac{\partial u_i}{\partial t} + \frac{\partial \left(u_j u_i\right)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j}\right) + f_i \tag{2.6}$$

The system of equations (2.5) and (2.6) forms the famous incompressible Navier-Stokes equations.

2.2 Turbulence modeling

The equations in section 2.1 describe in a deterministic way every kind of flow. However at high Reynolds number the flows exhibit a chaotic behavior called *turbulence*. The main features of turbulent flows are the strong dependency from the boundary conditions and the total absence of motion scale. This means that the flow has a chaotic behavior along all the spatial and temporal scales that need a statistical approach to be modeled. Actually, it is possible to directly integrate the complete equations using the DNS (*Direct Numerical Simulations*) but this requires a huge computational cost proportional to $Re_L^{9/4}$ (the subscript L means that the Re number is computed in the energy-containing range of the Kolmogorov's Energy spectra) which is too expansive for high Reynolds flow. A complete treatment about DNS and Kolmogorov's Energy spectra can be found in [6] and [7].
In general every scalar quantity in turbulent regime can be defined as $\phi = \langle\phi\rangle + \phi'$ where $\langle\phi\rangle$ is the time-averaged part and ϕ' is its instantaneous fluctuation. The time-averaged part is computed as $\langle\phi\rangle = \frac{1}{T}\int_T \phi(\mathbf{x}, t)dt$ where T is the time interval in which $\frac{1}{T}\int_T \phi'(\mathbf{x}, t)dt = 0$. Therefore, considering the three components of the velocity and the pressure it is possible to write:

$$u_i = \langle u_i\rangle + u_i' \tag{2.7}$$

$$p = \langle p\rangle + p' \tag{2.8}$$

Sobstituting (2.7) and (2.8) into the Navier-Stokes equations and applying the time-average operation leads to the *Reynolds-averaged Navier–Stokes equations* (RANS):

$$\frac{\partial \langle u_j\rangle}{\partial x_j} = 0 \tag{2.9}$$

$$\frac{\partial \langle u_i\rangle}{\partial t} + \frac{\partial \left(\langle u_i\rangle \langle u_j\rangle\right)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \langle u_i\rangle}{\partial x_j}\right) - \frac{\partial \left\langle u_i' u_j'\right\rangle}{\partial x_j} + \langle f_i\rangle \tag{2.10}$$

After the averaging operation it possible to notice how the averaged continuity equation (2.9) is basically unchanged, while in the averaged momentum equation (2.10) a new term, which dimensionally is a stress, appears $\rho\left\langle u_i' u_j'\right\rangle$. This is the so called

Reynolds stress tensor and it is formed by the product's average of the unwanted velocities fluctuations. It is a symmetric tensor and does not allow anymore the closure of the system and hence all the RANS models have as their main goal, the definition of this term. Different RANS models exist in literature and they can be divided in two big groups :

- *Turbulent-viscosity models* in which it is supposed a relation between the Reynolds stress tensor and the spatial derivative of the mean velocity components (*Boussinesq approximation*). This closure can be algebraic or differential, with one or more equations.

- *Reynolds-stress models* in which the model transport equation are solved for the individual Reynolds stresses $\langle u_i' u_j' \rangle$ and for the dissipation ε or another quantity (e.g ω) that provides a length or a time scale for the turbulence.

In this thesis work will be presented only the two main eddy-viscosity models the $k-\varepsilon$ model (section 2.2.1) and the $k-\omega$ SST model (section 2.2.2). A complete description about RANS modeling can be found in [8] and [9].
In compressible flow also the flow's density exhibit fluctuations, this strongly complicates the time-averaging operation especially in the momentum equation where the Reynolds-stress tensor originates from time averaging the product $\rho u_i u_j$ that appears in the convective acceleration. Clearly, a triple correlation involving $\rho' u_i' u_j'$ appears, thus increasing the complexity of establishing a suitable closure approximations. A simplification in the equations can be obtained introducing the *Favre averaging* operation, defined for a general scalar quantity ϕ by:

$$\tilde{\phi} = \frac{1}{\langle \rho \rangle} \int_T \rho(\mathbf{x}, t)\phi(\mathbf{x}, t)dt \tag{2.11}$$

Thus in terms of conventional Reynolds averaging, it is possible to say that:

$$\langle \rho \rangle \tilde{\phi} = \langle \rho \rangle \langle \phi \rangle + \langle \rho' \phi' \rangle \tag{2.12}$$

Using the Favre averaging it is customary to decompose the scalar variables in a mass averaged part $\tilde{\phi}$ and a fluctuating part ϕ''

$$\phi = \tilde{\phi} + \phi'' \tag{2.13}$$

To form the Favre average it is simply necessary to multiply by ρ both sides of (2.13) and do the time average operation described at the beginning of this section. After performing this operation and considering (2.12) it can be shown:

$$\langle \rho \phi'' \rangle = 0 \tag{2.14}$$

This allows a great mathematical simplification of the Favre averaged continuity (2.15), momentum (2.16) and energy (2.17) equations that assume a form really similar to their respective standard form (in order (2.1), (2.2), (2.3)) except for the presence of the fluctuating components of the variables.

$$\frac{\partial \langle \rho \rangle}{\partial t} + \frac{\partial \left(\langle \rho \rangle \tilde{u_j} \right)}{\partial x_j} = 0 \tag{2.15}$$

$$\frac{\partial\left(\langle\rho\rangle\,\tilde{u_i}\right)}{\partial t}+\frac{\partial\left(\langle\rho\rangle\,\tilde{u_j}\tilde{u_i}\right)}{\partial x_j}=-\frac{\partial\langle p\rangle}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\langle\sigma_{ij}\rangle-\left\langle\rho u_i''u_j''\right\rangle\right)+\langle\rho\rangle\,\tilde{f}_i \tag{2.16}$$

$$\frac{\partial\left(\langle\rho\rangle\,\tilde{E}\right)}{\partial t}+\frac{\partial\left(\langle\rho\rangle\,\tilde{u_j}\tilde{E}\right)}{\partial x_j}=-\frac{\partial\langle p\rangle\,\tilde{u_j}}{\partial x_j}+\frac{\partial\langle u_j\sigma_{ij}\rangle}{\partial x_j}-\frac{\partial\langle q_j\rangle}{\partial x_j}-\frac{\partial\left\langle u_j''p\right\rangle}{\partial x_j}-\frac{\left\langle\rho u_j''E''\right\rangle}{\partial x_j}+\langle S_E\rangle \tag{2.17}$$

$$\langle p\rangle=\bar{\rho}R\tilde{T}\text{ and }\tilde{e}=c_v\tilde{T} \tag{2.18}$$

As for the RANS modeling in compressible flow solvers the main aim is to compute a relation between the Favered-Averaged quantities and the fluctuating components in particular the Favre-averaged Reynolds stress tensor $\left\langle\rho u_i''u_j''\right\rangle$ that as in the incompressible case is a symmetric tensor. The interested reader can find a complete discussion about the modeling of compressible turbulent flows in [10].

2.2.1 $k-\varepsilon$ model

The $k-\varepsilon$ model belongs to the class of the turbulent-viscosity models. These models are all based on the Boussinesq approximation that the Reynolds stresses are given by:

$$\left\langle u_i'u_j'\right\rangle=\frac{2}{3}k\delta_{ij}-\nu_T\left(\frac{\partial\langle u_i\rangle}{\partial x_j}+\frac{\partial\langle u_j\rangle}{\partial x_i}\right)=\frac{2}{3}k\delta_{ij}-2\nu_T s_{ij} \tag{2.19}$$

Where $k=\frac{1}{2}\left\langle u_i'^2\right\rangle$ is the turbulent kinetic energy and $s_{ij}=\frac{1}{2}\left(\frac{\partial\langle u_i\rangle}{\partial x_j}+\frac{\partial\langle u_j\rangle}{\partial x_i}\right)$. Therefore, with this assumption, it is necessary to define or compute the *turbulent viscosity* scalar quantity ν_T in all the flow domain to close the system of equations.[1]

A problem of the models based on the Boussinesq approximation is that they always implies an isotropic assumption for the normal Reynolds stresses, indeed if we compute the tensor s_{ii}, letting $i=1,2,3$ while keeping $j=i$, it possible to notice considering the incompressibility constraint how $\left\langle u_{ii}'^2\right\rangle=\frac{2}{3}k$. This can lead to inaccurate results even for a simple 2-D flow.

The $k-\varepsilon$ model is a *two-equations model*, in which two additional transport equation are solved for the two turbulent quantities turbulent kinetic energy $k\,[\frac{m^2}{s^2}]$ and turbulent kinetic energy dissipation rate $\varepsilon\,[\frac{m^2}{s^3}]$. With this two quantities can be formed a length scale $\left(L=k^{3/2}\varepsilon\right)$, a time scale $\left(\tau=\frac{k}{\varepsilon}\right)$ and consequently the turbulent viscosity $\left(\nu_t=\frac{k^2}{\varepsilon}\right)$. The exact transport equation for k is: [2]

$$\frac{\partial k}{\partial t}+\langle u_j\rangle\frac{\partial k}{\partial x_j}=-\frac{\partial}{\partial x_j}\left(\frac{1}{2}\left\langle u_i'u_i'u_j'\right\rangle\right)-\frac{1}{\rho}\frac{\partial\left\langle u_j'p'\right\rangle}{\partial x_j}+\nu\frac{\partial^2k}{\partial x_j^2}-\left\langle u_i'u_j'\right\rangle\frac{\partial\langle u_i\rangle}{\partial x_j}-\nu\left\langle\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}\right\rangle \tag{2.20}$$

[1] Formally the quantity ν_T is a viscosity only from the dimensional point of view and it is called viscosity considering the analogy of the Boussinesq approximation and with the shear stress relations in a Newtonian fluid. The real viscosity is a physical property of the fluid and not of its motion.

[2] The exact transport equation for k can be derived in three steps, the first one is to obtain the transport equation for each velocity fluctuations u_i' subtracting 2.9 and 2.10 from 2.5 and 2.6, then multiplying the obtained equation for the velocity fluctuation u_i' and finally summing over i and applying the time averaging operation.

In (2.20) the budget for k is made of three terms: energy flux, (2.21), production (2.22) and energy dissipation (2.23)

$$\mathcal{T}_j \equiv \frac{1}{2}\left\langle u_i' u_i' u_j' \right\rangle + \frac{\left\langle u_j' p' \right\rangle}{\rho} - \nu \frac{\partial k}{\partial x_j} \tag{2.21}$$

$$\mathcal{P} \equiv -\left\langle u_i' u_j' \right\rangle \frac{\partial \left\langle u_i \right\rangle}{\partial x_j} \tag{2.22}$$

$$\varepsilon \equiv \nu \left\langle \frac{\partial u_i'}{\partial x_j} \frac{\partial u_i'}{\partial x_j} \right\rangle \tag{2.23}$$

In this way introducing the mean total derivative $\frac{\bar{D}(\bullet)}{\bar{D}t} \equiv \frac{\partial(\bullet)}{\partial t} + \left\langle u_j \right\rangle \frac{\partial(\bullet)}{\partial x_j}$ (2.20) can be written in a simplified way as:

$$\frac{\bar{D}k}{\bar{D}t} = -\frac{\partial\left(\mathcal{T}_j\right)}{\partial x_j} + \mathcal{P} - \varepsilon \tag{2.24}$$

As shown in (2.24) the terms $\frac{\bar{D}k}{\bar{D}t}$ and $\mathcal{P}$ are in closed form while $\mathcal{T}_j$ and ε need to be modeled in order to obtain a closed set of model equations. The energy flux is modeled with a gradient-diffusion hypothesis as:

$$\mathcal{T}_j = -\frac{\nu_T}{\sigma_k} \frac{\partial k}{\partial x_j} \tag{2.25}$$

Where σ_k is one of the five model constant. For the ε an exact equation can be derived but it is quite complex and involves other terms that do not allow the closure of the system. Therefore, the standard model equation for ε is best viewed as begin entirely empirical: it is

$$\frac{\bar{D}\varepsilon}{\bar{D}t} = \frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right) + C_{\varepsilon 1}\frac{\mathcal{P}\varepsilon}{k} - C_{\varepsilon 2}\frac{\varepsilon^2}{k} \tag{2.26}$$

Also in this case σ_ε, $C_{\varepsilon 1}$ and $C_{\varepsilon 2}$ are calibrated model's constant. To summarize the final equations to model the Reynolds stress tensor and close the Navier-Stokes equations with the $k-\varepsilon$ model are :

$$\left\langle u_i' u_j' \right\rangle = \frac{2}{3}k\delta_{ij} - \nu_T\left(\frac{\partial \left\langle u_i \right\rangle}{\partial x_j} + \frac{\partial \left\langle u_j \right\rangle}{\partial x_i}\right) = \frac{2}{3}k\delta_{ij} - 2\nu_T s_{ij} \tag{2.27}$$

$$\nu_T = C_\mu \frac{k^2}{\varepsilon} \tag{2.28}$$

$$\frac{\bar{D}k}{\bar{D}t} = \frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_k}\frac{\partial k}{\partial x_j}\right) + \mathcal{P} - \varepsilon \tag{2.29}$$

$$\frac{\bar{D}\varepsilon}{\bar{D}t} = \frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right) + C_{\varepsilon 1}\frac{\mathcal{P}\varepsilon}{k} - C_{\varepsilon 2}\frac{\varepsilon^2}{k} \tag{2.30}$$

With the five model constant

$$C_\mu = 0.09,\ C_{\varepsilon 1} = 1.44,\ C_{\varepsilon 2} = 1.92,\ \sigma_k = 1,\ \sigma_\varepsilon = 1.3 \tag{2.31}$$

2.2.2 $k-\omega$ SST model

As the $k-\varepsilon$ model the $k-\omega$ model is a two-equation model that rather than ε uses as second turbulent variable $\omega \equiv \frac{\varepsilon}{k}$ called turbulence frequency. Using ω definition its transport equation can be derived directly from (2.29) and (2.30) imposing $\sigma_k = \sigma_\varepsilon = \sigma_\omega$

$$\frac{\bar{D}\omega}{\bar{D}t} = \frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_\omega}\frac{\partial \omega}{\partial x_j}\right) + (C_{\varepsilon 1} - 1)\frac{\mathcal{P}\omega}{k} - (C_{\varepsilon 2} - 1)\,\omega^2 + \frac{2\nu_T}{\sigma_\omega k}\frac{\partial \omega}{\partial x_j}\frac{\partial k}{\partial x_j} \tag{2.32}$$

However, the real transport equation for ω in the standard $k-\omega$ model is:

$$\frac{\bar{D}\omega}{\bar{D}t} = \frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_\omega}\frac{\partial \omega}{\partial x_j}\right) + C_{\omega 1}\frac{\mathcal{P}\omega}{k} - C_{\omega 2}\omega^2 \tag{2.33}$$

In homogeneous turbulence 2.32 and 2.33 are identical considering $(C_{\varepsilon 1} - 1) = C_{\omega 1}$ and $(C_{\varepsilon 2} - 1) = C_{\omega 2}$. But, because most of the engineering's flows are inhomogeneous and do not allow the elimination of the differential term at the end of (2.32), in the standard transport equation for ω (2.33) the model's coefficients are different. About the transport equation for k, the two model are defined in the same way.
The $k-\omega$ shows a better behavior in the viscous sublayer compared to the $k-\varepsilon$ model, indeed in the viscous sublayer where $\varepsilon = 0$, ν_T, as defined in the $k-\varepsilon$ model, tends to diverge. This is why for the grids used with the $k-\varepsilon$ model the first grid point must be in the logarithmic layer with a $y^+ > 30$. In the $k-\omega$ model, the turbulent frequency in the viscous sublayer tends to infinity and it is possible to avoid the divergence of the turbulent viscosity with good predictions of turbulence phenomena like flow separation and reattachment near the wall. However, the $k-\omega$ model is very problematic in the free stream where both turbulent kinetic and turbulence frequency tend to zero. In this region the turbulent viscosity ν_T is indeterminate or infinite as ω tend to zero, so a small non zero value of ω needs to be specified. Unfortunately, results are dependent of the specified value of ω and this is a serious problem in aerospace and aerodynamics applications where free stream boundary conditions are used as a matter of routine. To get the best of both the $k-\varepsilon$ and $k-\omega$ model the $k-\omega$ SST (shear stress transport) model uses the $k-\omega$ model in the boundary layer and the $k-\varepsilon$ model in the free-stream. In the transition between the boundary layer and the free stream *blending functions* are used. The transport equation for k and ω are modified compared to the standard $k-\omega$ model, for both of them in the diffusive flux term is added the fluid's viscosity to better simulate low Reynolds flows, in the transport equation for ω are inserted the two blending functions F_1 and F_2 ,the dissipation term in the k transport equation is multiplied by the constant β^* and the production term for k features a limiter. To summarize the model equations of the $k-\omega$ SST model are:

$$\frac{\bar{D}k}{\bar{D}t} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_T}{\sigma_k} + \nu\right)\frac{\partial k}{\partial x_j}\right] + \tilde{\mathcal{P}}_k - \omega k \beta^* \tag{2.34}$$

$$\frac{\bar{D}\omega}{\bar{D}t} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_T}{\sigma_\omega} + \nu\right)\frac{\partial \omega}{\partial x_j}\right] + \mathcal{P}_\omega - C_{\omega 2}\omega^2 + 2\,(1 - F_1)\,\frac{\sigma_{\omega 2}}{\omega}\frac{\partial \omega}{\partial x_j}\frac{\partial k}{\partial x_j} \tag{2.35}$$

$$\nu_T = \frac{a_1 k}{max\,(a_1\omega, SF_2)} \tag{2.36}$$

$$\tilde{\mathcal{P}}_k = min(\mathcal{P}_k, 10 \cdot \beta^* k\omega) \text{ with } \mathcal{P}_k = \nu_T \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right) \frac{\partial \langle u_i \rangle}{\partial x_j} \tag{2.37}$$

$$\mathcal{P}_\omega = C_{\omega 1} S^2 \text{ with } S = \sqrt{2 s_{ij} s_{ij}} \tag{2.38}$$

$$a_1 = \frac{5}{9} \tag{2.39}$$

$$F_2 = tanh \left\{ \left[max \left(\frac{2\sqrt{k}}{\omega \beta^* y}, \frac{500\nu}{y^2 \omega} \right) \right]^2 \right\} \tag{2.40}$$

$$\beta^* = \frac{9}{100},\ C_{\omega 1} = 0.44,\ C_{\omega 2} = 0.0828,\ \sigma\omega = 0.5,\ \sigma\omega 2 = 0.856 \tag{2.41}$$

$$F_1 = \tanh \left\{ \left[min \left(max \left(\frac{\sqrt{k}}{\omega \beta^* y}, \frac{500\nu}{y^2 \omega} \right), \frac{4\sigma_{\omega 2} k}{C D_{k\omega} y^2} \right) \right]^4 \right\} \tag{2.42}$$

$$C D_{k\omega} = max \left(2 \frac{\sigma_{\omega_2}}{\omega} \frac{\partial \omega}{\partial x_j} \frac{\partial k}{\partial x_j}, 10^{-10} \right) \tag{2.43}$$

This last equations show how difficult and empirical is the definition of a good method to the closure of the RANS equations. About the $k - \omega$ SST model, it is possible to say that it show a low sensibility at the boundary condition, it works well at low Reynolds number as far as in adverse pressure gradients and flow separation problems. These features make it one of the most used RANS models. Most equation's derivations in sections 2.2.1 and 2.2.2 are taken from [11], while for the interested reader a full description of the two mentioned turbulence models can be found in [12] and [13].

2.2.3 Large-eddy simulation (LES)

In large eddy simulation the larger three dimensional unsteady turbulent motions are directly solved, whereas the effects of the smaller scales motions are modeled. In terms of computational cost they lie between the Reynolds stress model and DNS and the separation between the two scales of motion is done through a filtering operation. To better understand the concept behind the filtering operation, it can be useful to consider the Kolmogorov's spectra of isotropic turbulence (Figure 2.1 [14]). The filtering operation cuts the energy spectra and in this way the eddies below a certain wavenumber are completely resolved, while the small eddies of high wave number are modeled, these modeled eddies are commonly called small sub-grid scales (SGS). The cutoff curve is a function of the adopted filter and in physical space this means that the actual velocity can be decomposed (2.44) in a filtered quantity $\bar{u_i}$ and a modeled sub-grid quantity u_i'.

$$u_i = \bar{u_i} + u_i' \tag{2.44}$$

The quantity $\bar{u_i}$ is defined through the use of the filtering function $G(\mathbf{x}, \mathbf{r})$ by the convolution integral

$$\bar{\mathbf{u}}(\mathbf{x}, t) = \int_{-\infty}^{\infty} G(\mathbf{x}, \mathbf{r})\, \mathbf{u}(\mathbf{x} - \mathbf{r}, t)\, d\mathbf{r} \tag{2.45}$$

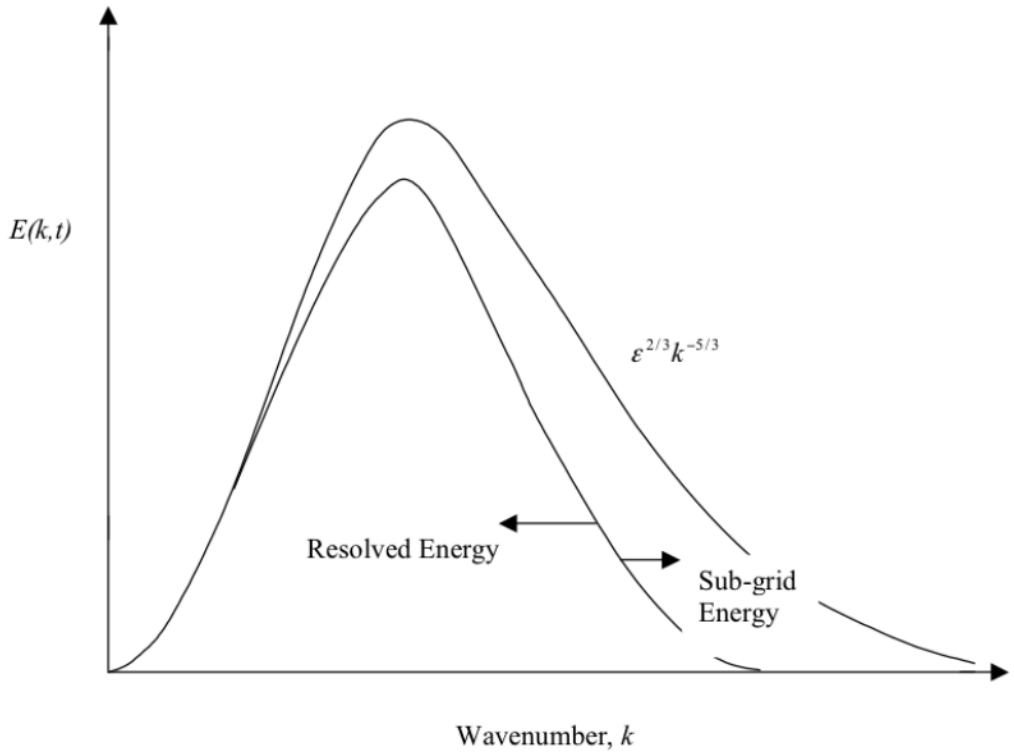


Figure 2.1: Effects of filtering operation on isotropic turbulence

that must satisfy the normalization condition:

$$\int_{-\infty}^{\infty} G(\mathbf{x}, \mathbf{r})\, d\mathbf{r} = 1 \tag{2.46}$$

There are different kind of filter functions and their mathematical definition is different in the physical and spectral space which are connected through the Laplace transform operation.Table 2.1 [15] shows the most common filters in both spectra and physical space. For the filters Δ is the filter length and among them the most commonly used are the box filter and the sharp spectral filter. The box filter has a very simply explanation in physical space, it computes $\bar{\mathbf{u}}(\mathbf{x})$ as the average of $\mathbf{u}(\mathbf{x'})$ in the interval $\mathbf{x} - \frac{1}{2\Delta} < \mathbf{x}' < \mathbf{x} + \frac{1}{2\Delta}$. The sharp spectral filter has instead a very clear explanation in spectral space. For this filter, all wave numbers below the cut off value κ_c are resolved while all wave numbers above the cut-off are modeled.
The derivation of the LES equation is done in this section for the hypothesis of incompressible flow, however the filtering operation can be applied also for compressible flow (see Appendix A). Before proceeding with the derivation of the filtered Navier-Stokes equation, it is important to define some properties of the filtering operation. First of all, unlike the averaging operation in RANS for a generic scalar quantity ϕ, we have that $\bar{\bar{\phi}} \neq \bar{\phi}$ and that $\bar{\phi}' \neq 0$. Second, since the convolution integral in the filtering operation involves the product of two functions, the operation commutes only for a spatially uniform filter $\frac{\overline{\partial \phi}}{\partial x} = \frac{\partial \bar{\phi}}{\partial x}$.

Name	**Filter function**	**Transfer function**
General	$G(\mathbf{r})$	$\widehat{G}(\kappa) = \int_{-\infty}^{\infty} e^{i\kappa\mathbf{r}} G(\mathbf{r})$
Box	$\frac{1}{\Delta} H\left(\frac{1}{2}\Delta - \lvert\mathbf{r}\rvert\right)$	$\frac{\sin\left(\frac{1}{2}\kappa\Delta\right)}{\frac{1}{2}\kappa\Delta}$
Gaussian	$\left(\frac{6}{\pi\Delta^2}\right)^{1/2} \exp\left(-\frac{6\mathbf{r}^2}{\Delta^2}\right)$	$\exp\left(-\frac{\kappa^2\Delta^2}{24}\right)$
Sharp spectral	$\frac{\sin\left(\frac{\pi\mathbf{r}}{\Delta}\right)}{\pi\mathbf{r}}$	$H\left(\kappa_c - \lvert\kappa\rvert\right)\ ;\ \kappa_c \equiv \frac{\pi}{\Delta}$
Cauchy	$\frac{a}{\pi\Delta\left[\left(\frac{\mathbf{r}}{\Delta}\right)+a^2\right]}\ ;\ a = \frac{\pi}{24}$	$\exp(-a\Delta\lvert\kappa\rvert)$
Pao		$\exp\left[-\frac{\pi^{2/3}}{24}\left(\Delta\lvert\kappa\rvert\right)^{4/3}\right]$

Table 2.1: Common LES filters

Considering a spatially uniform filter and applying the filtering operation to the Navier-Stokes equations leads to:

$$\frac{\partial \bar{u}_j}{\partial x_j} = 0 \tag{2.47}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \overline{u_i u_j}}{\partial x_j} = \nu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} \tag{2.48}$$

Since the product $\overline{u_i u_j} \neq \bar{u}_i \bar{u}_j$, using (2.44) and applying again the filtering operation to (2.48) gives:

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = \nu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.49}$$

The term :

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j \tag{2.50}$$

in (2.49) is called *sub-grid stress tensor* and it is the analogous of the Reynolds-stress tensor of the RANS. As for the RANS the main goal of LES models is to define this tensor in terms of the filtered velocities. However, it must be underlined, that the fields $\bar{\mathbf{u}}(\mathbf{x}, t)$, $\bar{p}(\mathbf{x}, t)$ and τ_{ij} are random three dimensional and unsteady even in case of homogeneous flow. Moreover, the sub-grid stress tensor depends on the specification of the type and width of the filter. In this work will be presented only the two most common models to the closure of the system of equations the

Smagorisky Model and the *Dynamic Smagorisky model*, many other models are discussed in literature. The interested reader can have a look at [16], [17] and [18]

Smagorisky model

The main hypothesis of the Smagorisky model is that the residual stress tensor is a scalar multiple of the rate of strain tensor. This is a very weak assumption in fact, the sub-grid stress tensor correlates very poorly with the rate of strain tensor. This is obvious for incompressible fluids where the trace of the strain tensor is zero, which implies that at least one term on the diagonal is positive and one is negative, while the diagonal terms of the sub-grid stress tensor are all grater than zero. For these reasons, it is not possible to find a scalar that will correctly relate the sub-grid stress tensor to the rate of strain tensor. To help this realizability problem, the trace of the sub-grid stress tensor is added to the sub-grid stress tensor making the diagonal positive. However, this make the problem ill posed because there are an infinite number of sub-grid stresses traces that will satisfy the expression. To guarantee consistency in the equation, the trace term is also added to the filtered pressure to give a pseudo-pressure. Defining $k_r \equiv \frac{1}{2}\tau_{ii}$ *(Residual Kinetic Energy)* the Smagorisky model takes the form:

$$\bar{p} \equiv \bar{p} + \frac{2}{3}k_r \tag{2.51}$$

$$\tau_{ij} = \nu_T \overline{S_{ij}} + \frac{2}{3}k_r\delta_{ij} \tag{2.52}$$

With the filtered rate of strain tensor $\overline{S_{ij}}$ and the eddy viscosity ν_T defined as:

$$\overline{S_{ij}} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.53}$$

$$\nu_T = -2C_s\Delta^2|\overline{S}| \tag{2.54}$$

$|\overline{S}|$ is the magnitude of the rate of strain tensor and it is computed as:

$$|\overline{S}| = \left(2\bar{S_{ij}}\bar{S_{ij}}\right)^{1/2} \tag{2.55}$$

The parameter C_s is a user-defined coefficient that may vary significantly depending on the flow and grid resolution. This is another weakness of the Smagorisky model that for complex flow requires an a priori knowledge of the C_s value that sometimes is not available. Furthermore, for this kind of flows the coefficient may not be appropriate for the whole domain at all times.

Dynamic Smagorisky model

The dynamic modeling has been developed by [19], with this technique instead of using an universal model coefficient, the model coefficient is dynamically determined as a function of space and time from the resolved field. This approach is based on an assumed scaling between resolved and subgrid scales and a mathematical identity that arises. The main advantage of these models is that they do not require an a priori knowledge of the flow to set the flow coefficient. The dynamic modeling involves filters of different widths, the *Grid Filter* and the *Test Filter*. Generally

the grid filter has a width proportional to the grid spacing, while the test filter has generally a width twice the one of the grid filter. Denoting with $\overline{\mathbf{u}}$ the filtering operation with the grid filter, with $\widehat{\mathbf{u}}$ the one with the test filter and considering that both are spatially uniform it is possible to write:

$$\widehat{\overline{\mathbf{u}}} \equiv \int \mathbf{u}(\mathbf{x}-\mathbf{r})G(|\mathbf{r}|;\bar{\Delta})G(|\mathbf{r}|;\widehat{\Delta})d\mathbf{r} \equiv \int \overline{\mathbf{u}}(\mathbf{x}-\mathbf{r})G(|\mathbf{r}|;\widehat{\Delta})d\mathbf{r} \equiv \int \mathbf{u}(\mathbf{x}-\mathbf{r})G(|\mathbf{r}|;\widehat{\bar{\Delta}}) \tag{2.56}$$

It directly follow the decomposition:

$$\mathbf{u} = \widehat{\overline{\mathbf{u}}} + \left(\overline{\mathbf{u}} - \widehat{\overline{\mathbf{u}}}\right) + \mathbf{u'} \tag{2.57}$$

$\overline{\mathbf{u}} - \widehat{\overline{\mathbf{u}}}$ can be interpreted as the smallest resolve motions by of grid of spacing $\bar{\Delta}$ or equivalently the largest motions not resolved by a grid of spacing $\widehat{\bar{\Delta}}$. As for the single filtered equations it is possible to define a sub-grid stress tensor based on the double filtering operation:

$$\mathcal{T}_{ij} \equiv \widehat{\overline{u_i u_j}} - \widehat{\bar{u}_i}\widehat{\bar{u}_j} \tag{2.58}$$

Subtrating (2.58) to the test filtered (2.50) leads to the famous Germano identity

$$\mathcal{L}_{ij} \equiv \mathcal{T}_{ij} - \widehat{\tau}_{ij} = \widehat{\bar{u}_i \bar{u}_j} - \widehat{\bar{u}_i}\widehat{\bar{u}_j} \tag{2.59}$$

This identity is extremely powerful because it relates the unknown stress tensor at the two scales to $\mathcal{L}_{ij}$ (the so called *Leonard stress Tensor*) which is known in terms of $\bar{\mathbf{u}}$. Therefore, all dynamics models are based on the definition of $\mathcal{T}_{ij}$ and τ_{ij} in terms of the filtered velocity, and then using the Germano identity obtain an adequate coefficient C_s for the specific flow. The dynamic Smagorisky model follows the same assumptions of the simple Smagorisky model in the definition of τ_{ij} and $\mathcal{T}_{ij}$. Therefore using (2.52), (2.53), (2.54) and (2.55) it is possible to define:

$$\tau_{ij} \equiv -2C_s\Delta^2\overline{|S|}\overline{S_{ij}} + \frac{1}{3}\tau_{kk}\delta_{ij} \tag{2.60}$$

$$\mathcal{T}_{ij} \equiv -2C_s\widehat{\bar{\Delta}}^2\widehat{\overline{|S|}}\widehat{\overline{S_{ij}}} + \frac{1}{3}\mathcal{T}_{kk}\delta_{ij} \tag{2.61}$$

Then defining

$$M_{ij} \equiv 2\widehat{\Delta}^2\widehat{\overline{|S|}\overline{S_{ij}}} - 2\widehat{\bar{\Delta}}^2\widehat{\overline{|S|}}\widehat{\overline{S_{ij}}} \tag{2.62}$$

The deviatoric part of the Leonard stress tensor can be modeled as:

$$\mathcal{L}_{ij} - \frac{1}{3}\mathcal{L}_{kk} = C_s M_{ij} \tag{2.63}$$

(2.63) can be used to obtain the best value of the coefficient C_s because both $\mathcal{L}_{ij}$ and M_{ij} are known in terms of $\overline{\mathbf{u}}$. However, C_s can not be determined in order to match exactly the nine components of the two tensors, but as shown by [20] the mean square error between the two tensor is minimized by the algebraic equation:

$$C_s = \frac{M_{ij}\mathcal{L}_{ij}}{M_{kl}M_{kl}} \tag{2.64}$$

The coefficient C_s obtained in this way can be positive or negative, a positive value means that the energy flows from the resolved to the sub-grid scales while a negative coefficient implies the contrary. This short summary about the mathematical modeling of LES has been mainly done consulting [21] and [15].

2.2.4 Detached-Eddy Simulation (DES)

The Detached eddy simulation is an hybrid approach between LES and RANS. The definition of this approach is given by [22]:

A Detached-Eddy simulation is a three-dimensional unsteady solution using a single turbulence model, which functions as a sub-grid scale model in regions where the grid density is fine enough for a large-eddy simulation, and as a Reynolds-averaged model in region where it is not.

Therefore in a Detached-Eddy Simulation according to the provided mesh, the model chooses turbulent the length scale as:

$$L_{DES} = min(L_{RANS}, C_{DES}\Delta) \tag{2.65}$$

Where C_{DES} is a modeling parameter empirically determined and Δ is the filter width taken as maximum dimension of the local grid cell:

$$\Delta = max(\Delta_x, \Delta_y, \Delta_z) \tag{2.66}$$

The advantages of DES are that they are capable to treat high Reynolds number flows with massive separation, without requiring the huge computational cost of a true wall-bounded layer LES. Indeed LES requires a very fine mesh resolution near the walls to model in a good way the small eddies of this flow region. However, the drawback of this model is in the so called *grey area* where $L_{RANS} \approx C_{DES}\Delta$. Here the solution is not neither pure RANS or pure LES and the model needs to convert from fully modeled turbulence (RANS) to mostly resolved turbulence with mass separation (LES). This results in a weakened eddy viscosity, but not too weak to allow LES eddies to form, resulting in lower Reynolds stress levels compared to those provided by the RANS model. These eddyes are generally extremely elongated and with unphysically long time scales [23].

The $k-\omega$ SST DES turbulence model

The $k-\omega$ SST DES model is a DES modification of the RANS $k-\omega$ SST model. In this model the transport equations for k and ω are exactly the same of the $k-\omega$ model (Eqs. (2.34) - (2.43))[3] the only difference is in the dissipation term of the k transport equation. This term is multiplied by the term F_{DES} which reads as:

$$F_{DES} = max\left(\frac{L_T}{C_{DES}\Delta}, 1\right) \tag{2.67}$$

In (2.67) L_T is the turbulent length scale computed according to $k-\omega$ SST model as $L_T = \sqrt{k}/(\beta^*\omega)$, as described in (2.66) Δ is the largest side of a cell at the present point in the grid and C_{DES} is the empirical constant of the $k-\omega$ SST DES model equal to 0.61 . When the grid is fine enough the term F_{DES} grows, this reduces k and consequently ν_T, allowing the solution to go unsteady and to be treated as a

[3]The transport equation can be written independently of $\overline{u_i}$ or $\langle u_i \rangle$ because the averaging or the filtering operation are not actually done in the solver. This is possible because the form of the filtered and averaged Navier-Stokes equations is exactly the same and hence also the algorithm to solve the equations

pure LES. To prevent the model to go unsteady in the *grey zone* the term F_{DES} can be further modified as:

$$F_{DDES} = max\left(\frac{L_t}{C_{DES}\Delta}(1 - F_S), 1\right) \tag{2.68}$$

Where F_S is a blending function chosen as either F1 or F2. Because they assume a value close to 1 in the boundary layer, this will grantee to the solution to not go unsteady near the wall, avoiding the entrance in the grey zone. With these feature the model is called DDES (Delayed Detached Eddy Simulation).

Chapter 3

Introduction to OpenFOAM

OpenFOAM (Operation Field and Manipulation) is a an open source software mainly created for the CFD analysis and it's main aim is to solve partial differential equation through the finite volume method. Strictly speaking it is not a a real software, but a library written in C++ which can create executable files called *applications*. Inside this library are already compiled a huge amount of applications which cover different physical phenomena like complex fluid with chemical reactions, heat transfer and turbulence models. The applications are divided in two categories : *solvers* and *utilities*. The former are coded to solve continuum mechanics problems, the latter are used for the pre- and post- processing of the simulation data. One of the strong points of OpenFOAM is that it's source code is completely available to the user, who can modify and change it. This helps the creation of personalized applications in a small amount of time compared to their creation from scratch.

3.1 OpenFOAM's structure

The structure of OpenFOAM can be summarized in Figure 3.1 [24] and during the post-processing phase it is very important to use third-party software to view the simulation results. Between these software the most used is certainly ParaView (it has been used also for this thesis work) that interacts with OpenFOAM through the utility *paraFoam*. To launch a simulation in OpenFOAM it is necessary to define a folder which contains all the necessary files to handle the simulation. This folder has inside it other three subfolders (Figure 3.2):

- **constant** which depending on the analyzed problem and the used solver, contains the *thermophysical properties* file that specifies the thermodynamical and physical properties of the fluid, the *transport property* file in which are defined the fluid's transport properties, the *turbulence properties* file with the used turbulence model and the *PolyMesh* folder where are stored all the mesh data and the simulation boundary conditions.

- **system** in which are stored the files for the mesh generation (an example is the *blockMeshDict* file), the *controlDict* file where it is possible to define the settings of the simulation like the simulation time, the time-step, the write interval and the used solver. The methods for the discretization of the solved equations are specified in the *fvSchemes* file, while in the *fvSolutions* file are

specified the solvers for the discretize equations. Depending on the case problem, other files can be included in the system directory, like the *decomposeParDict* file to decompose the domain simulation on several processors and run the simulation in parallel, the *snappyHexMeshDict* that reconstructs the mesh around a 3D-body starting from the description of its surfaces in a *.stl* file and other files for the pre-processing (ex. *topoSetDict* and *setFieldsDict*) and the post-processing (ex. *sampleDict*).

- **0 directory** which is the first of the time directories that will be created by the solver during the simulation. In this directory are specified the initial condition for each physical quantity of the simulation like velocity, temperature and pressure.

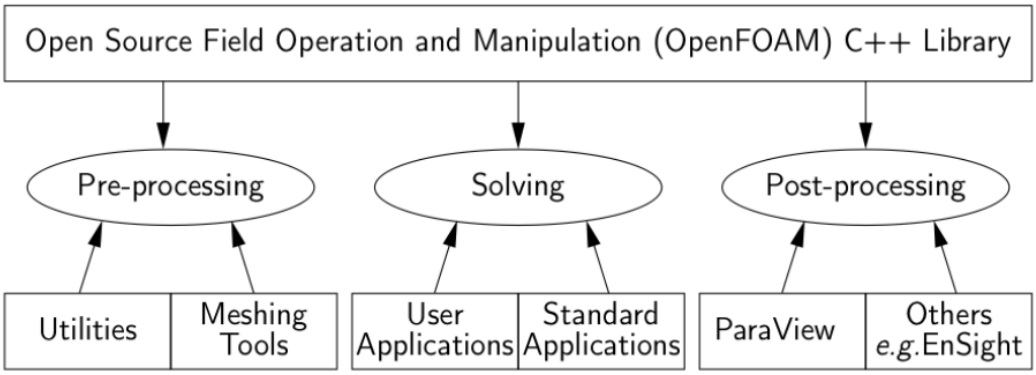


Figure 3.1: OpenFOAM structure

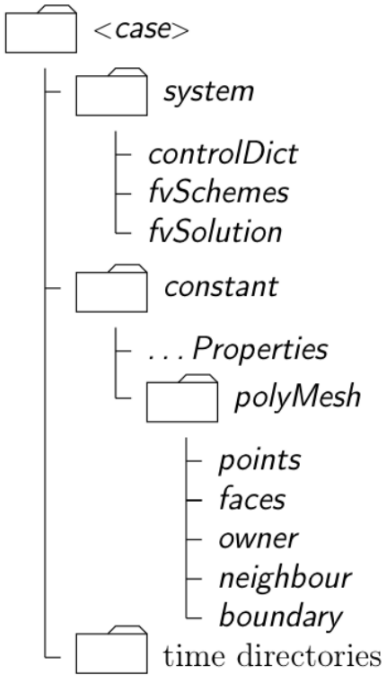


Figure 3.2: Simulation's folder structure

3.2 The finite volume method

As stated in the previous paragraph the solution of partial differential equation in OpenFOAM is based on the finite volume method, therefore it is appropriate to summarize the theory behind this method to better understand the discretization schemes in the *fvSchemes* file. Considering the general transport equation for a scalar quantity ϕ:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho\phi u_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\rho\Gamma\frac{\partial\phi}{\partial x_j}\right) + S_\phi \tag{3.1}$$

With the hypothesis of continuum functions in the whole domain, the operations of integration and derivation can be commuted and applying the volume integral on each of the mesh cells it is possible to write:

$$\frac{\partial}{\partial t}\int_V (\rho\phi)\,dV + \int_V \frac{\partial}{\partial x_j}(\rho u_j\phi)\,dV = \int_V \frac{\partial}{\partial x_j}\left(\rho\Gamma\frac{\partial\phi}{\partial x_j}\right)dV + \int_V S_\phi dV \tag{3.2}$$

Using the Gauss theorem and denoting with ∂V the surfaces of each cell of the mesh and with n_j the versor normal to each surface:

$$\frac{\partial}{\partial t}\int_V (\rho\phi)\,dV + \oint_{\partial V} (\rho u_j\phi)\,n_j dS = \oint_{\partial V}\left(\rho\Gamma\frac{\partial\phi}{\partial x_j}\right)n_j dS + \int_V S_\phi dV \tag{3.3}$$

The aim of the finite volume method is to solve (3.3) for each cell of the mesh (control volume), considering as unknown the value of ϕ at the cell center and approximating the fluxes through the cell's surfaces using the nearby cells. In this way it is possible to have the value of ϕ in all the cells of the domain and have an approximation of the scalar field that is obviously more accurate increasing the number of cells.The approximation of the surfaces flux and the time discretization can be done in several ways depending on which term in the equation is discretized (convective term, gradient term, diffusive term etc.), the desired order of accuracy and the time available for the simulation.

3.2.1 Time discretization

In the finite volume method the time derivative term as well as the source term is taken as piecewise constants over the control volume.

$$\frac{\partial}{\partial t}\int_V (\rho\phi)\,dV = V_{CV}\frac{\partial}{\partial t}(\rho\phi) \tag{3.4}$$

$$\int_V S_\phi dV = V_{CV} S_\phi \tag{3.5}$$

The discretization of the time derivative term determines the way the algorithm update the solution in time. The main discretization schemes are the implicit Euler, the explicit Euler and the Crank Nicholson method.Considering a generically partial differential equation:

$$\frac{\partial(\rho\phi)}{\partial t} = f(\phi,\psi) \tag{3.6}$$

They can be explained using the discretization:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = f\left(\theta\phi^{n+1} + (1-\theta)\phi^n, \psi^n\right) \tag{3.7}$$

The variable ψ is set as explicit as it is not solved in the equation. If $\theta = 0$ the discretization is called explicit Euler and it is first order accurate [4]. For $\theta = 1$ the discretization is called implicit Euler and it is also first order accurate. For $\theta = 1/2$ it is the Crank-Nicholson method which is between the implict and explicit Euler but it is second order accurate. More advanced schemes like Runge-Kutta are also available for time discretization, they can reach higher order of accuracy, but are more computationally expansive.

3.2.2 Convective discretization

The convective term can be approximated by:

$$\oint_{\partial V} (\rho u_j \phi)\, n_j dS = \sum_k \int_{S_k} \rho u_j \phi n_j S_k \tag{3.8}$$

Where k is the number of faces of the control volume. For a Cartesian grid in three dimension, a control volume is an hexahedron and it has six faces (Figure 3.3).

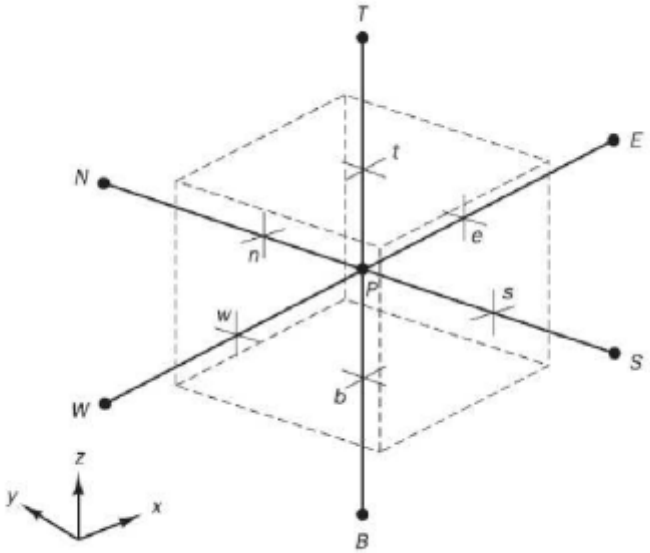


Figure 3.3: Control volume in three dimension with neighboring nodes

[3]The explicit Euler method is called explicit because it can computes ϕ^{n+1} knowing only the solution at the time $t = n$, while the Crank-Nicolson and the implicit Euler are both implicit because ϕ^{n+1} can not be determined only from the solution at time $t = n$ but also at time $t = n+1$. This involves the solution of a system of algebraic equations. It must be also added that the explicit Euler and the Cranck-Nicholson method are not unconditionally stable, while this is true for the implicit Euler.

Because the value of ϕ is not known at the volume's faces, an interpolation has to be made between the cell nodes. The treatment of this terms is one of the major challenges in CFD and there are many options to compute it. In this work will be presented only the most common, the *Centered scheme* and the *Upwind scheme*

Centered scheme

The centered scheme interpolates linearly the value of ϕ at the control volume faces using the near control volume that has the same common face . This approach is unlikely to be used in CFD applications as the interpolation schemes are not bounded and do not fulfill trasportiveness requirements.

Upwind sheme

A very simple solution is the Upwind schemes, a first order accurate scheme which evaluates the value of ϕ on a face of the control volume following the main direction of the flow. For example, considering a 2D mesh and the east face (e) of one its cells (P),(Figure 3.4) the upwind schemes can be expressed as:

$$\phi_e = \begin{cases} \phi_P \text{ if } (\mathbf{u}\cdot\mathbf{n})_e > 0 \\ \phi_E \text{ if } (\mathbf{u}\cdot\mathbf{n})_e < 0 \end{cases} \tag{3.9}$$

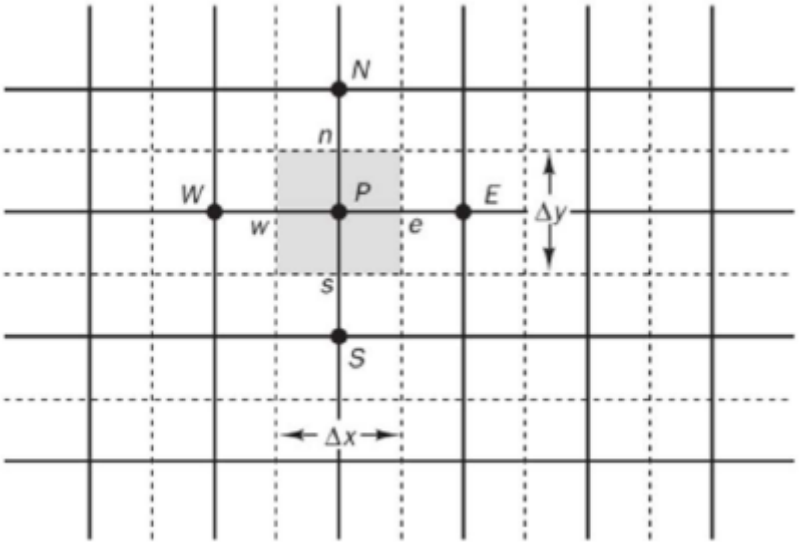


Figure 3.4: Cartesian notation for a control volume in two dimensions

Where the subscript E refers to the east neighbor cell of P. The issue with this scheme is that when the flow is not aligned with the grid a false diffusion error is introduced in the solution. Other versions of the Upwind scheme are the LUDS (linear upwind scheme) and the QUICK (quadratic upwind scheme), they are based on the same principle but they use interpolation between nodes to increase the order of accuracy.The LUDS is second order accurate while the QUICK is third order accurate.

3.2.3 Viscous discretization

The viscous term can be approximated by:

$$\oint_{\partial V} \left(\rho \Gamma \frac{\partial \phi}{\partial x_j} \right) n_j dS = \sum_k \int_{S_k} \rho \Gamma \frac{\partial \phi}{\partial x_j} n_j dS_k \tag{3.10}$$

This requires the calculation of the gradients at the surfaces of the control volumes, considering again a 2D mesh and supposing that it necessary to compute the gradient at the east face (e) of its control volume, this can be approximated using the relation:

$$\left(\frac{\partial \phi}{\partial x} \right)_e \approx \frac{\phi_E - \phi_P}{x_E - x_P} \tag{3.11}$$

Where again, P denotes the center of the control volume and E the center of the control volume near to P which has in common face e. This operation needs to be done for all the faces of the control volume in all the three directions, then all the gradients need to be multiplied for their respective faces and summed to approximate the value of the integral in (3.10).

3.2.4 Gradient discretization

Even if in the general transport equation for a scalar quantity ϕ are not present gradients terms, this is not true for the equations of fluid motion (2.1, 2.2 and 2.3). These terms can be easily approximated by

$$\oint_{\partial V} \frac{\partial \phi}{\partial x_i} n_j dS = \sum_k \int_{S_k} \phi n_j dS_k \tag{3.12}$$

Where the values of ϕ are calculated at the surfaces of the control volumes using an interpolation technique.

3.2.5 Available discretization schemes in OpenFOAM

In the *fvSchemes* dictionary of OpenFOAM, the set of terms for which numerical schemes must be specified are subdivided in:

- **timeScheme** : first and second derivative e.g. $\partial/\partial t$ and $\partial^2/\partial t^2$
- **gradSchemes** : gradient terms e.g. $\partial\phi/\partial x_i$
- **divSchemes** : convective terms e.g. $\partial\left(u_j\phi\right)/\partial x_j$
- **laplacianSchemes** : diffiusive terms e.g. $\frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right)$
- **interpolationSchemes** : cell to face interpolation of values
- **snGradSchemes** : component of gradient normal to a cell face
- **wallDist** : distance to wall calculation where required e.g. in the wall functions of the $k-\omega$ SST model.

timeSchemes

Euler	First order, bounded, implicit
localEuler	Local-time step, first order, bounded, implicit
CrankNicholson	Second order bounded implicit
backward	Second order implicit
steadyState	No solving for time derivatives

Table 3.1: OpenFOAM's time schemes

Table 3.1 summarize the available time schemes in OpenFOAM, for the CrankNicholson method the parameter ψ needs to be specified. For $\psi = 1$, the normal Crank-Nicholson is used, whereas if $\psi = 0$ it correspond to the Euler scheme.

gradSchemes

Gauss <interpolationScheme>	Second order, Gaussian integration
leastSquares	Second order, least square
Cubic	Third order, least square
cellLimited <gradScheme>	Second order implicit
faceLimited <gradScheme>	No solving for time derivatives

Table 3.2: OpenFOAM's gradient schemes

Table 3.2 summarize the gradient schemes in OpenFOAM. The Gauss entry specifies the standard finite volume discretization (3.2.4). If the <interpolationScheme> is specified as linear it means that the values at the CV faces are calculated using a linear interpolation between the center of the nearby cells.The other available interpolation schemes are CubicCorrection (cubic scheme) and midPoint (linear interpolation with symmetric weighting). The cellLimited and the faceLimited options limits the gradient such that when cells values are extrapolated to faces using the calculated gradient, the faces values do not fall outside the bounds of values in surroundings cells. This requires the specification of a limiting coefficient between 0 and 1. 1 guarantees boundedness and 0 applies no limiting. In general 1 is used as coefficient. In the least square the values at the CV's surfaces is approximated using the least square distance calculation using all the neighbor cells. The Third scheme is only used on regular meshes for DNS simulations. In general for this terms the default scheme is set as Gauss linear.

divSchemes

This schemes include the discretization of all the convective terms in the equations. The keyword identifier for the convective terms are usually of the form div (phi,..) , where phi denotes the volumetric flux of the velocity through the faces of the CVs. It is better to subdivide the convective terms in two categories, the convective term for the velocity (div (phi,U)) and the convective terms for the scalar quantities (div(phi,k) , div(phi,e) etc.). Table 3.3 summarize the Gauss discretization schemes for the velocity's convection

Gauss linear	Second order unbounded
Gauss linear upwind	Second order, upwind-biased, unbounded
Gauss LUST	Blended scheme 75% linear/ 25% linearUpwind
Gauss limitedLinear	Schemes that limits towards upwind on the regions of rapid changing gradient
Gauss upwind	first-order bounded generally too inaccurate to be recommended

Table 3.3: OpenFOAM's divergence schemes

The LUST discretization needs the specification of the velocity gradient while the limitedLinear discretization needs a coefficient between 0 and 1 to specify the type of limitation. 1 is the strongest limiting tending to upwind, while 0 is the weakest tending to linear. For the advection of the velocity there are also specialized *V-schemes* that computes a limits for the velocity based on its most rapidly changing component. They can be linear or upwind based. For the convection of scalar quantities the available options are the same and the Gauss limited case is specified without the final V. Moreover, for these quantities the limitedLinear and the upwind schemes are more used since there is more interest in the boundedness of the solution. An additional appearance in the transport of scalar quantities is finally the *VanLeer* scheme which is another limiting scheme less strong than the option *limitedLinear*. Due to the large amount of options for the convective terms, the default *divScheme* is set to none.

lapalacianSchemes

For the Laplacian terms the Gauss scheme is the only choice of discretization and requires a selection of both an interpolation scheme for the diffusion coefficient (linear,cubic or midPoint) and a surface normal gradient scheme. Therefore the general syntax for the Laplacian terms is : Gauss <interpolationScheme> <snGradScheme>. In general the default specification is Gauss linear corrected. A detailed explanation of how the surface normal gradients are evaluated is presented in the next paragraph.

snGradSchemes

The surface normal gradients are very important for the approximation of the Laplacian terms. They allow to computes the gradient of a physical quantity normal to a cell face using the CV centers of the 2 cells that the face connects. Table 3.4 shows the availble option for these schemes.

Corrected	Explicit non-orthogonal correction
Limited corrected	Limited non-orthogonal correction
Orthogonal	Simple approximation for Cartesian grids
Uncorrected	No non-orthogonal correction

Table 3.4: OpenFOAM's surface gradient schemes

The orthogonal scheme is the one described by (3.10). However this requires a regular mesh, typically aligned with the Cartesian co-ordinate system which does not

occur for meshes of engineering geometries. Therefore to maintain a second order accuracy, an explicit non-orthogonal correction can be added to the orthogonal component, forming the *corrected* scheme. The non-orthogonality correction increases as the angle α between the cell-cell vector and the face normal vector increases and as α tends to 90° the correction can be so large to slow down the simulation time and get the solution unstable. Because of this, the *limited corrected* scheme introduce a coefficient ψ between 0 and 1 to define a blended scheme between the corrected and uncorrected ones. 1 corresponds to the corrected scheme, while 0 corresponds to the uncorrected scheme. The uncorrected and corrected schemes are recommended for meshes with very low non-orthogonality and for meshes with maximum orthogonality above 70° the *limited* option may be required. All the details about the finite volume method and the various discretization techniques can be found in [25] while for the details on OpenFOAM discretization techniques the reader can have a look at [24]

3.3 Solution of the discretized equations

After the equations have been discretized they form a system of equations of the form

$$A\mathbf{x} = \mathbf{Q} \tag{3.13}$$

Because for large geometries these systems are too big to solve directly (e.g using Gaussian elimination or LU decomposition), OpenFOAM uses iterative procedures to solve them.

3.3.1 Iterative methods

The main idea about iterative methods consists in setting up a sequence of vectors $\mathbf{x}^n$ that *converges* to the exact solution $\mathbf{x}$ so that :

$$\lim_{n\to\infty} \mathbf{x}^n = \mathbf{x}. \tag{3.14}$$

In this way after n iterations it is possible to say that:

$$A\mathbf{x}^n = \mathbf{Q} - \mathbf{r}^n \tag{3.15}$$

And subtracting it from (3.13) it is possible to derive a relation between the iteration error $\mathbf{e}^n = \mathbf{x} - \mathbf{x}^n$ and the residual $\mathbf{r}^n$:

$$A\mathbf{e}^n = \mathbf{r}^n \tag{3.16}$$

At convergence the $\mathbf{e}$ and $\mathbf{r}$ must be zero and this can be reached forming an iterative scheme of the form :

$$M\mathbf{x}^{n+1} = N\mathbf{x}^n + \mathbf{B} \tag{3.17}$$

Since at convergence by definition $\mathbf{x}^{n+1} = \mathbf{x}^n = \mathbf{x}$ the relations between the matrix M, N and B and the original system (3.13) can be expressed as :

$$A = M - N \text{ and } \mathbf{B} = \mathbf{Q} \tag{3.18}$$

Or more generally

$$PA = M - N \text{ and } \mathbf{B} = P\mathbf{Q} \tag{3.19}$$

Where P is a non-singular *pre-conditioning* matrix. Different kind of iterative methods exist in literature but in this work will be summarized only the ones used in OpenFOAM. The interested reader can find further information in [26] and [27].

Conjugate Gradient Methods

These methods are mainly used to solve systems of non-linear equations like the Navier-Stokes equations. The main idea behind these methods is to convert the original system of equations into a minimization problem of the form:

$$F = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{Q} \tag{3.20}$$

For positive definite matrices, find the solution of the system (3.13) is equivalent to find the minimum of F for all the x_i . However, most matrices associated with problems in fluid dynamics are not symmetric or positive defined, and a way to convert the original system into a minimization problem, that does not require the positive definiteness, is to take the sum of squares of all the equations. The best known method for seeking the minimum of a function is the *steepest descend*, where the function F is thought to be a surface in a (hyper)-space of the same dimension of $\mathbf{x}$. Starting from an initial guess $\mathbf{x}_0$, that represents a point in the hyper-space, the gradient of F is computed in this point allowing to find the steepest downward path on the surface. Then the lowest point of the path is found and by construction it has a lower value on F compared to $\mathbf{x}_0$. The new value is then chosen as new starting point and the procedure is repeated until convergence. To speed up the convergence the *conjugate gradient* method is based on the remarkable discovery that it is possible to minimize a function with respect to several directions simultaneously while searching in one direction at a time. Considering two directions $\mathbf{p}_1$ and $\mathbf{p}_2$ to minimize F in the $\mathbf{p}_1 - \mathbf{p}_2$ plane it must be verified that :

$$\mathbf{p}_1 A\mathbf{p}_2 = 0 \tag{3.21}$$

This property is akin orthogonality and the vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are said to be conjugate respect matrix A, which gives the method its name. This property can be extended to any number of directions and each new search direction is required to be conjugate with all the previous ones.
The rate of convergence of this method depends on the *condition number* $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ where λ_{max} and λ_{min} are the largest and smallest eigenvalues of the matrix. Because the condition number of matrices that arise in CFD problems is approximately the square of the maximum number of grid points in any direction, it is necessary to multiply A by a preconditioning matrix to increase the rate of convergence. The preconditioning take the form of:

$$C^{-1}AC^{-1}C\mathbf{x} = C^{-1}\mathbf{Q} \tag{3.22}$$

Applying the *conjugate gradient method* to the new problem formulation the following algorithm results. In the description $\mathbf{r}^k$ is the residuals of the kth iteration, $\mathbf{p}^k$ is the kth search direction, $\mathbf{z}^k$ is an auxiliary vector and α_k and β_k are parameters used in constructing the new solution, residual and search direction. Here is reported a summary of the algorithm:

- Initialize by setting: $k = 0$, $\mathbf{x}^0 = \mathbf{x}_{in}$, $r^0 = \mathbf{Q} - A\mathbf{x}_{in}$, $\mathbf{p}^0 = 0$, $s_0 = 10^{30}$
- Advance the counter: $k = k + 1$
- Solve the system: $M\mathbf{z}^k = \mathbf{r}^{k-1}$
- Calculate:
  $s^k = \mathbf{r}^{k-1} \cdot \mathbf{z}^k$
  $\beta^k = s^k / s^{k-1}$
  $\mathbf{p}^k = \mathbf{z}^k + \beta^k \mathbf{p}^{k-1}$
  $\alpha^k = s^k / (\mathbf{p}^k \cdot A\mathbf{p}^k)$
  $\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha^k \mathbf{p}^k$
  $\mathbf{r}^k = \mathbf{r}^{k-1} - \alpha^k A\mathbf{p}^k$
- Repeat until convergence.

The algorithm involves solving a system of linear equations at the first step. The matrix involved is $M = C^{-1}$ where C is the preconditioning matrix which is in fact never constructed. The most common choice for M is the incomplete Cholesky factorization of A which is very easy to invert.

Biconjugate Gradients and CGSTAB

The conjugate gradient method it is applicable only to symmetric systems , to apply the method to systems of equations that are not symmetric it is necessary to convert an asymmetric problem to a symmetric one. To make this, the system can be decompose into two subsystems. The first is the original system, the second involves the transpose matrix $A^T\mathbf{y} = 0$ and it is irrelevant. When the pre-conditioned conjugate gradient method is applied to this system, the following method, called *biconjugate gradients* results:

- Initialize by setting: $k = 0$, $\mathbf{x}^0 = \mathbf{x}_{in}$, $\mathbf{r}^0 = \mathbf{Q} - A\mathbf{x}_{in}$, $\overline{\mathbf{r}}^0 = \mathbf{Q} - A^T\mathbf{x}_{in}$, $\mathbf{p}^0 = \overline{\mathbf{p}}^0 = 0$, $s_0 = 10^{30}$
- Advance the counter: $k = k + 1$
- Solve the systems: $M\mathbf{z}^k = \mathbf{r}^{k-1}$, $M^T\overline{\mathbf{z}}^k = \overline{\mathbf{r}}^{k-1}$
- Calculate:
  $s^k = \mathbf{z}^k \cdot \overline{\mathbf{r}}^{k-1}$
  $\beta^k = s^k / s^{k-1}$
  $\mathbf{p}^k = \mathbf{z}^k + \beta^k \mathbf{p}^{k-1}$
  $\overline{\mathbf{p}}^k = \overline{\mathbf{z}}^k + \beta^k \overline{\mathbf{p}}^{k-1}$
  $\alpha^k = s^k / (\overline{\mathbf{p}}^k A\mathbf{p}^k)$
  $x^k = x^{k-1} + \alpha^k \mathbf{p}^k$
  $\mathbf{r}^k = \mathbf{r}^{k-1} - \alpha^k A\mathbf{p}^k$
  $\overline{\mathbf{r}}^k = \overline{\mathbf{r}}^{k-1} - \alpha^k A^T\overline{\mathbf{p}}^k$
- Repeat until convergence

This algorithm requires twice as much effort per iteration as the the standard conjugate gradient method but it converges in about the same number of iterations. Other variant of the *biconjugate* gradient method are the CGS (conjugate gradient squared) algorithm; the CGSStab(CGS stabilized) and the GMRES, another version of the CGSStab. All these algorithms works for symmetric and non-symmetric matrices and both structured and un-structured grids. Here is finally reported the CGSTAB algorithm:

- Initialize by setting: $k = 0$, $\mathbf{x}^0 = \mathbf{x}_{in}$, $\mathbf{r}^0 = \mathbf{Q} - A\mathbf{x}_{in}$, $\mathbf{u}^0 = \mathbf{p}^0 = 0$
- Advance the counter $k = k + 1$ and calculate:
  $\beta^k = \mathbf{r}^0 \cdot \mathbf{r}^{k-1}$
  $\omega^k = (\beta^k \gamma^{k-1})/(\alpha^{k-1}\beta^{k-1})$
  $\mathbf{p}^k = \mathbf{r}^{k-1} + \omega^k(\mathbf{p}^{k-1} - \alpha^{k-1}\mathbf{u}^{k-1})$
- Solve the system : $M\mathbf{z} = \mathbf{p}^k$
- Calculate:
  $\mathbf{u}^k = A\mathbf{z}$
  $\gamma^k = \beta^k/(\mathbf{u}^k \cdot \mathbf{r}^0)$
  $\mathbf{w} = \mathbf{r}^{k-1} - \gamma^k\mathbf{u}^k$
- Solve the system : $M\mathbf{y} = \mathbf{w}$
- Calculate :
  $\mathbf{v} = A\mathbf{y}$
  $\alpha^k = (\mathbf{v} \cdot \mathbf{r}^k)/(\mathbf{v} \cdot \mathbf{v})$
  $\mathbf{x}^k = \mathbf{x}^{k-1} + \gamma^k\mathbf{z} + \alpha^k\mathbf{y}$
  $\mathbf{r}^k = \mathbf{w} - \alpha^k\mathbf{v}$
- Repeat until convergence

Multigrid Methods

The basis of multigrid methods is that in iterative methods the rate of convergence depends on the eigenvalues of the iteration matrix. In particular, the eigenvalue(s) with largest magnitude determines how rapidly the solution is reached and the eigenvector(s) associated with this eigenvalue(s) determines the spatial distribution of the iteration error. Specifically, some iterative methods (ex. Gauss-Seidel) remove after a few iterations the rapidly varying component of the iteration error that becomes a smooth function of the spatial coordinate. This means that it is possible to compute the update (an approximation to the iteration error) on a coarse grid, reducing the iteration cost. As an example on a 2d-grid twice as coarse as the original one, iterations cost 1/4 as much. Therefore, the procedure in a multigrid method is the following:

- On the fine grid, perform iterations with a method that gives a smooth error (smoother)

- Once the error is smooth and the most rapidly component of the iteration error have been removed compute the residuals on the fine grid
- Restrict the residuals on the coarse grid
- Perform iterations on the coarse grid until the iteration error is again smooth
- Using the iteration error computed on the coarse grid correct the one on the fine grid using an interpolation technique
- Update the solution on the fine grid
- Repeat the entire procedure until the residual is reduced to the desired level

This is a very general procedure and it is possible to continue to use coarser grids to improve the rate of convergence. Moreover, multigrid is more a strategy than a particular method and a lot of parameters (smoother, number of iterations on each grid, interpolation schemes etc.) are selected more or less arbitrarily.
This short summary about iterative methods for the solution of system of equations have been done consulting [28]

3.3.2 OpenFOAM equations solvers

The available equations solvers in OpenFOAM are:

- **PCG/PBiCGStab**: Stabilized preconditioned (bi-)conjugate gradient, for booth symmetric and asymmetric matrices.
- **PCG/PBiCG**: preconditioned (bi-)conjugate gradient, with PCG for symmetric matrices, PBiCG for asymmetric matrices.
- **smoothSolver**: solver that uses a smoother.
- **GAMG**: generalized geometric-algebraic multi-grid.
- **diagonal**: diagonal solver for explicit system.

As it is clear from the name the PCG/PBiCGStab and the PCG/PBiCG belongs to the class of the conjugate gradient methods while the smooth solver and the GAMG are multigrid methods. The diagonal solver is the only direct method an it uses the LU factorization for the solution of a system of equations.
An equation solver must be specified for each simulation's variable in the *fvSchemes* file, together with the desired tolerance and relative tolerance to stop the iterations in the solution of the system. Generally, in transient simulation the relative tolerance is set to 0 to force the solution to converge to the solver tolerance in each time step. Depending on the solver used in the simulation the *fvSchemes* file contains also other parameters involving the algorithm used by the simulation solver (ex. number of external loop for the PIMPLE algorithm) and the relaxation factors in case of steady state simulations involving the SIMPLE algorithm.

Chapter 4

Solver selection, set-up and validation

The entire thesis work has as main aim the simulation of a jet flow exiting from a CFM-56 aircraft engine in flight condition, the selected solver in OpenFOAM is *rhoPimpleFoam*. This is a transient solver for turbulent flows of compressible fluids and it is a *pressure-based* solver that uses the PIMPLE algorithm to solve the Navier-Stokes equations. Even if the flow is in steady-state conditions, it has been decided to use a transient solver because at flight conditions the flow exiting from the nozzle is at sonic conditions, therefore during the simulations, if the mesh is enough refined, some shocks can be captured without allowing the solution to diverge. Moreover, the *rhoPimpleFoam* solver differently from the other steady-state solvers like *rhoSimpleFoam* can support the $k-\omega$ SST DES turbulence model that is used in the final part of the work.

4.1 The PIMPLE algorithm

The PIMPLE algorithm is a mix between the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) and the PISO algorithm (Pressure Implicit with Splitting of Operators). These algorithms were borne to solve incompressible flow simulations and are all pressure-based, which means that the velocity and pressure field are solved together through the so called *pressure-velocity coupling*. Even if the simulated flow is clearly compressible, it is useful to have a brief introduction to the incompressible version of these algorithms to better understand their modified compressible version.
The problem of the incompressible Navier-Stokes equations is that there is not an independent pressure equation, but it appears in the gradient form in the momentum equation. Therefore a relation that directly relates the pressure to the velocity field is needed an it can be obtained applying the divergence operator to the continuity equation. Considering the incompressibility constraints, this leads to the Poisson equation:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{(\partial \rho u_i u_j)}{\partial x_j}\right] \tag{4.1}$$

where the outer derivatives of the pressure inside the brackets must be discretized in the same way they are discretized in the momentum equations; while the outer derivatives, which come from the continuity equation must be approximated in the

way they are discretized in the continuity equation. In all the three algorithms the iterations within one time-steps are called *outer iterations*, they are performed in an *outer loop* in which the coefficients and the source matrix of the discretized equations are updated. The operations performed on linear systems with fixed coefficients are called instead *inner iterations* and they occur in the so called *inner loop*. Starting with the SIMPLE algorithm the first step is to solve the discretized momentum equation considering the pressure field and the source term of the previous iterations:

$$A_P^{u_i} u_{i,P}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{4.2}$$

here P is the index of the computed node, l is the index of the center of the cell adjacent to p, Q is the matrix of the source terms, A the matrix of the velocity coefficients referred to node P and m the a generic index to identify a generic iteration. In a compact way this system of equations can be seen as $[A]\mathbf{u}^{m*} = \mathbf{b}_{m-1} - \bigtriangledown \mathbf{p}_{m-1}$. The velocities u^{m*} obtained at node P can be expressed as:

$$u_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_l^{ui} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{4.3}$$

or in a more compact form as:

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{4.4}$$

These velocities do not satisfy the continuity equation, which is why they carried an asterisk. The next step is therefore to introduce a small correction to the velocity and pressure field inside the inner loop, denoting with the apex m the velocity field that satisfy the continuity equation it is possible to write:

$$u_i^m = u_i^{m*} + u' \text{and } p^m = p^{m-1} + p' \tag{4.5}$$

Sobstituting (4.5) in (4.3) allows to introduce a relation between u' and p'

$$u'_{i,P} = \tilde{u'}_{i,P} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p'}{\delta x_i}\right)_P \tag{4.6}$$

where

$$\tilde{u'}_{i,P} = -\frac{\sum_l A_l^{u_i} u'_{i,l}}{A_P^{u_i}} \tag{4.7}$$

Then considering the discretized continuity equation

$$\frac{\delta\left(\rho u_i^m\right)}{\delta x_i} = 0 \tag{4.8}$$

with the use of (4.6) it is possible to introduce an equation that directly relates p' with the velocities u_i^{m*}

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)\right]_P = \left[\frac{\delta(\rho u_i^{m*})}{\delta x_i}\right]_P + \left[\frac{\delta(\rho \tilde{u_i}')}{\delta x_i}\right]_P \tag{4.9}$$

which is basically the discretized Poisson equation (4.1) expressed in terms of the velocity and pressure corrections. In the SIMPLE algorithm the velocity corrections

$\tilde{u}_i'$ are unknown and hence neglected, therefore p' is expressed as a only function of u_i^{m*}. Then the corrected pressure is entered again in (4.3) in order to obtain a new velocity field u_i^{m*} and repeat the procedure until the pressure correction falls below a given tolerance and the velocity field satisfy both continuity and momentum equation. Because $\tilde{u_i'}$ is neglected the SIMPLE algorithm converges slowly and it is used mainly for steady-state simulations. Furthermore, to avoid instabilities relaxation factors α_P and α_u are introduced in the computation of p^m and u_i^{m*}.

$$p^m = p^{m-1} + \alpha_p p' \tag{4.10}$$

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \alpha_u \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{4.11}$$

To speed up the convergence the PISO algorithm after neglecting $\tilde{u}_i'$ and computed the pressure correction p' using (4.6), computes $u_{i,P}'$ as:

$$u_{i,P}' = -\frac{1}{A_P^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_P \tag{4.12}$$

Allowing the computation of $\tilde{u}_i'$ using (4.7).
Then defining the second velocity corrections as:

$$u_{i,P}'' = \tilde{u}_{i,P}' - \frac{1}{A_P^{u_i}} \left(\frac{\delta p''}{\delta x_i} \right)_P \tag{4.13}$$

and substituting in the discretized continuity equation (4.8) allows to write the second pressure correction equation:

$$\frac{\delta}{\delta x_i} \left[\frac{\rho}{A_P^{u_i}} \left(\frac{\delta p''}{\delta x_i} \right) \right]_P = \left[\frac{\delta(\rho \tilde{u}_i')}{\delta x_i} \right]_P \tag{4.14}$$

So what basically the PISO algorithm makes more compared the SIMPLE algorithm is to add an inner loop to correct a second time the pressure and the velocity. This speed up the convergence allowing the use of this algorithm also in transient simulations. Following the procedure described by Equations (4.12) - (4.14), further corrector steps can be created increasing both the convergence and the computational cost of the algorithm. As said at the beginning of this section the PIMPLE algorithm merge the PIMPLE and the SIMPLE algorithm allowing the user to choose the number of inner loop (number of corrector steps that can be constructed) and outer loop (changing of the coefficient matrix $[A]$ and the source term $\mathbf{b}$) at each time step of the simulation. In a very schematic way the PIMPLE algorithm can be summarized by the following pseudo-code:

```
for t = to.....tn
  while n outer loop <= n max outer loop and Tol >= maxTol
  .assemble the matrix of the discretized momentum equation
  .solve discretized momentum equation
  .assemble the matrix of the discretized Poisson equation
  .solve Poisson equation for pressure correction
  .correct pressure and velocity field
    for n inner loops
```

```
    .assemble the matrix of the discretized Poisson equation
    .solve Poisson equation for pressure correction
    .correct pressure and velocity field
    end
  .solve turbulence and other transport quantities
  .update tolerance
 end
end
```

In compressible flows the continuity equation is not only a matter of momentum, indeed because the density change with the temperature it necessary to introduce a step for the density correction inside the solution process. As in the incompressible SIMPLE algorithm the first step is compute u^{m*} using the momentum equation and the density ρ^{m-1} of the previous iteration. This allows the computation of the mass fluxes that a the beginning will not satisfy the continuity equation for each face of the control volume. Considering the east face S_e of the control volume (Figure 3.3) the mass flux through the face can be written as:

$$\dot{m}_e^m = \left(\rho^{m-1} + \rho'\right)_e \left(u_n^{m*} + u_n'\right)_e S_e \tag{4.15}$$

Exapnding the above equation, the mass flow correction is defined as:

$$\dot{m}_e' = \left(\rho^{m-1} u_n'\right) S_e + \left(u_n^{m*} \rho'\right) S_e + \left(\rho' u_n'\right)_e \tag{4.16}$$

The last term on the right end sides of the equations has a lower order of magnitude and converges faster compared to the others therefore it is neglected. As for the incompressible case, sobstituting the corrected max flux in the momentum equation allows to identify a relation between the mass flow correction and the pressure correction p'. Again for the sake of simplicity considering the momentum balance on the east face of the CV:

$$\left(\rho^{m-1} u_n'\right) S_e + \left(u_n^{m*} \rho'\right) S_e = \left(\rho^{m-1} S_e\right) \left(\frac{1}{A_P}\right)_e \left(\frac{\delta p}{\delta n}\right)_e \tag{4.17}$$

It is now necessary to establish a relation between the pressure correction p' and the density correction, this can be easily achieved considering the equation of state for a perfect gas (2.4) and expanding it via a Taylor series expansion:

$$\rho|_{p^n + p'} = \rho|(p^n) + \frac{\delta \rho}{\delta p} p' = \rho * + \rho' \Rightarrow \rho' = \frac{\delta \rho}{\delta p} p' = \frac{1}{RT} p' = C_\rho p' \tag{4.18}$$

Where T is the fluid temperature computed solving the energy equation using the values of the previous iterations. Neglecting the velocity correction as for the incompressible case it is now possible to identify a relation that relates directly $\dot{m}'$ and p' :

$$\dot{m}_e' = \left(\rho^{m-1} S_e\right) \left(\frac{1}{A_P}\right)_e \left(\frac{\delta p}{\delta n}\right)_e + \left(\frac{C_\rho \dot{m}^*}{\rho^{m-1}}\right)_e p_e' \tag{4.19}$$

Where $\dot{m^*}$ is defined as the product between u_n^{m*} and ρ^{m-1}. Writing (4.19) for all the six faces of the CV and substituting in the continuity equation allows to find the pressure correction p' and finally with (4.19) together with (4.16) the velocity correction u_n'. This is basically the SIMPLE part of the PIMPLE algorithm for compressible flows, in the PISO part second corrections are introduced before update the matrix coefficients of the discretized equations. The decription of the SIMPLE,PISO and PIMPLE algorithm has been done consulting [29] and [30]

4.2 The SET-UP case

To set-up the solver it has been decided to use as validation case the near-sonic jet flow of the NASA Langely research center [1]. The simulation involves a nozzle (Acoustic research Nozzle 2, o ARN-2) with radius 1 inch (25.4 mm) and it is compared with the experimental data of [31] and [32].This flow is near $Ma = 1$ at the nozzle exit as for the CFM-56 engine, but it exits into quiescent air, while for the aircraft case there is a strong co-flow of $252 m/s$. Moreover, this is an isothermal case while for the CFM-56 nozzle the outlet temperature is approximately of 600 K. Because this is an axisymmetric case periodic rotating boundary conditions have been used and below is reported a scheme with the mesh geometry and the physical initial condition:

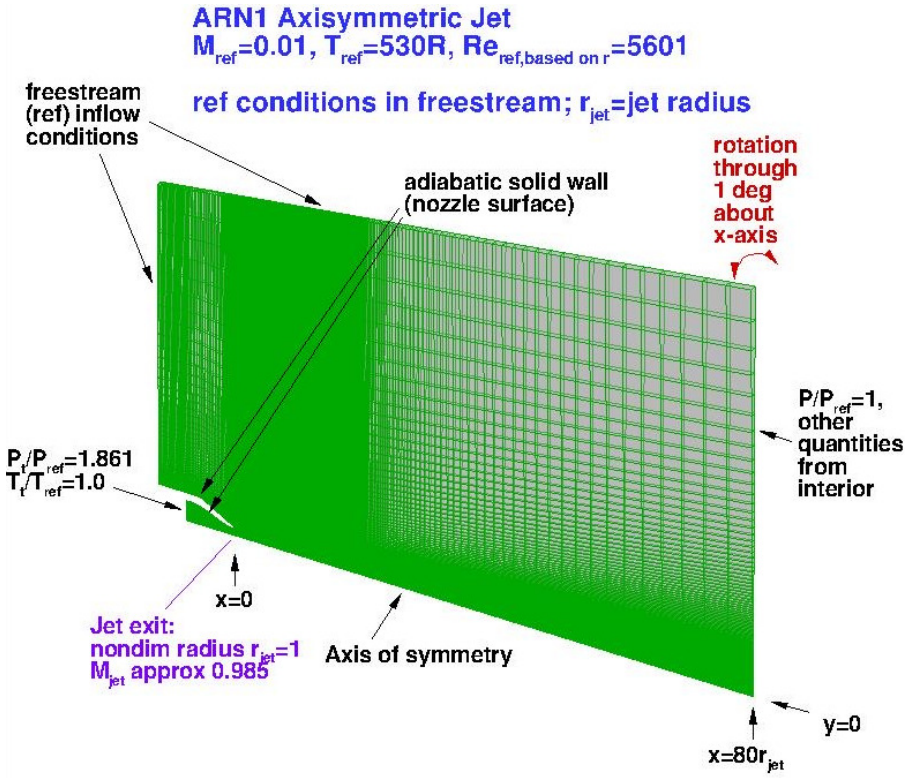


Figure 4.1: NASA set-up and validation case

4.2.1 Mesh Generation using the *BlockMesh* utility

Even if the grids used for the NASA case are available for the download, they are in the plot3D format which is not easily convertible in OpenFOAM and does not allow to export the various geometry patches for the definition of the boundary conditions. For these reasons it has been decided to use the OpenFOAM utility *BlockMesh* to generate a mesh similar to the one used by the NASA case. The principle of *BlockMesh* is to decompose the domain geometry in a set of hexahedral blocks. The edges of these block can be straight lines , circles or spline. Each block is defined by 8 vertices and the local reference system inside the block must be right handed. The reference system is defined by the order in which the vertices are presented in the blocks, the type of edges are by default straights lines and to change

them it is necessary to define a list named *edges*. Each item of the list contains a keyword specifying the type of curve connecting two vertices of the block (ex arc, spline, polyLine etc.), the numbers which identify the vertices to be connected and a series of interpolation points from which the edge needs to pass. Do define a block it is necessary to define three entries:

- **Vertex numbering** In which are identified the vertices composing the blocks. They are always preceded by the word *hex* that identifies the shape of the block which is always an hexaedron.
- **Number of cells** Where are defined the number of cells in each of the three directions of the block x_1 , x_2 and x_3.
- **Cell expansions ratios** This is a very important parameter and it defines how much the cells expand in each direction of the block. It is defined as :

$$Ex.\ ratio = \frac{\delta e}{\delta s} \tag{4.20}$$

  where δe is the the width of the cell at the end of one edge of the block and δs is the width of the cell at the beginning of the edge. The expansion happens in geometric progression and the common ratio q is related to the expansion ratio through the relation :

$$q = \sqrt[n-1]{Ex.\ ratio} \tag{4.21}$$

  where n is the number of cells of the block's edge.

After the definition of the blocks with their number of cells and expansion ratios, it necessary to define the boundaries of the mesh, this is given in a list named *boundary*. The boundary is broken into patches (regions), where each patch in the list has its name as the keyword. The keyword is chosen by the user and it will be used in the 0 directory to define the initial conditions on that boundary for the start of the simulation. The patch information is then contained in a sub-dictionary with:

- **type**, which can be a generic patch on which are applied some boundary condition, or a geometric condition. In this case it is necessary to define the patch *wall* for the nozzle boundaries, the patch *symmetryPlane* for the axis of symmetry of the wedge geometry and the patch *wedge* for the two side faces of the wedge planes. It is this last geometry patch that defines the periodic rotating boundary conditions
- **faces** which is a list of block faces that make up the patch. Each face is identified by a list of 4 vertex numbers. It is important that looking from inside the block and starting with any of the vertices, the face must be traversed in a clock wise direction to define the other vertices.

For further details about the mesh generation using *BlockMesh* it is possible to see [24] while the complete script for the mesh generation of the NASA case can be seen in Appendix B.
About the geometric dimensions, the NASA case and the OpenFOAM test case are the same. This is not true for the number of blocks and the number of cells in the

mesh. In particular the mesh constructed with *BlockMesh* in the OpenFOAM case has been divided in six blocks and it is a bit less refined compared to the one used by NASA (see Figures 4.2 - 4.4 and Table 4.1).

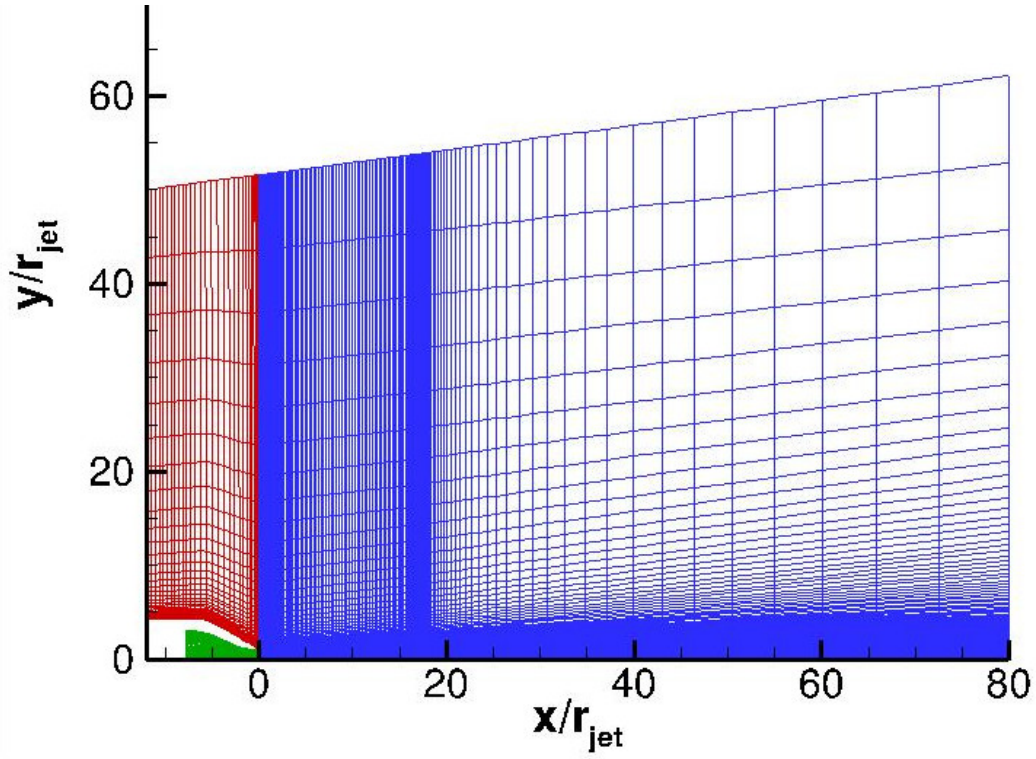


Figure 4.2: Near sonic jet case geometry dimensions

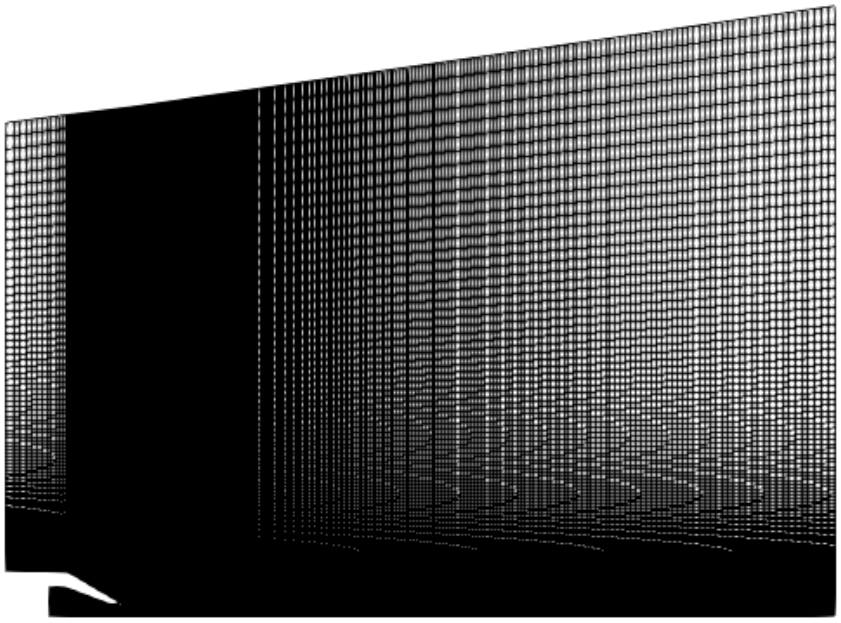


Figure 4.3: Near sonic jet case OpenFOAM's mesh

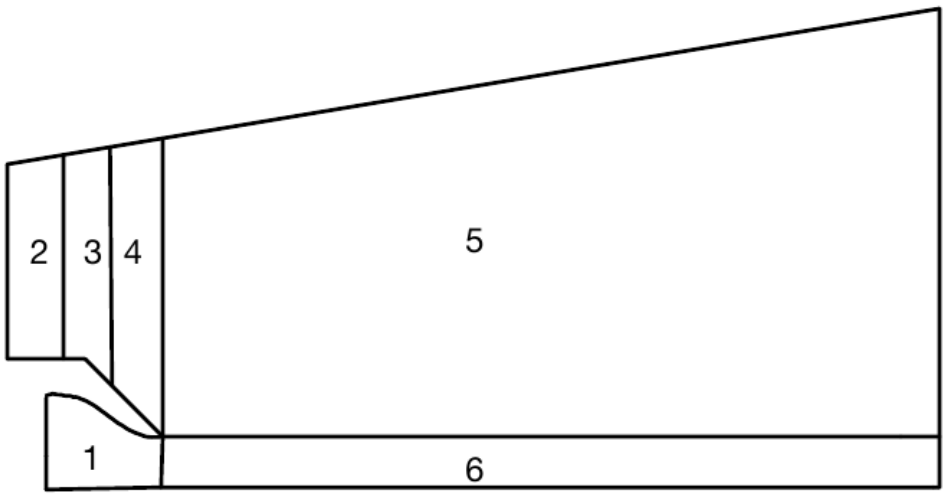


Figure 4.4: Near sonic jet case OpenFOAM's mesh blocks (figure not in scale)

	N. cells x	N. cells y	N. cells z	Ex. ratio x	Ex. ratio y
Block 1	97	97	1	1	1
Block 2	11	168	1	0.5	150
Block 3	3	168	1	0.5	150
Block 4	46	168	1	0.435	150
Block 5	257	168	1	8.6	150
Block 6	257	97	1	8.6	1

Table 4.1: Near sonic jet case OpenFOAM's cells per block

4.2.2 Selection of the turbulence model and of the thermophysical properties

To run the simulation it has been decided to use the $k-\omega$ SST turbulence model. This model as described in section 2.2.2 is well suited for almost every kind of flow and it is used in the SST-V version for the NASA test case. The only difference between the standard $k-\omega$ SST model implemented in OpenFOAM and the V version used in the NASA case is in the production term $\mathcal{P}$ (4.22) of both the transport equation for k and ω which is defined in terms of the vorticity magnitude $\Omega = \sqrt{2W_{ij}W_{ij}}$ with $Wij = \frac{1}{2}\left(\frac{\partial\langle u_i\rangle}{\partial x_j} - \frac{\partial\langle u_j\rangle}{\partial x_i}\right)$ [33]

$$\mathcal{P} = \nu_T \Omega^2 - \frac{2}{3} k \delta_{ij} \frac{\partial \langle u_i \rangle}{\partial x_j} \tag{4.22}$$

The fluid exting from the nozzle is air and it is treated as a single mixture perfect gas with molar mass 28.9 $kg/Kmol$ and constant heat capacity at constant pressure $C_p = 1005\ J/kg$. The viscosity μ is considered function of the temperature T with the well known *Southerland* relation:

$$\mu = \frac{A_s\sqrt{T}}{1 + T_s/T} \tag{4.23}$$

With $A_s = 1.458 \cdot 10^{-6} \left[Pa \cdot s \cdot K^{-1/2}\right]$ and $T_s = 110.4\,[K]$ as Southerland's coefficients for air. These property are evaluated through the OpenFOAM *hePsiTermo* model which uses as variable for the energy equation the fluid internal energy e. Below is reported the *thermophysicalProperties* file defined in the constant directory of the OpenFOAM case.

```
/*---------------------------*- C++ -*----------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       sutherland;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture // air at room temperature (293 K)
{
    specie
    {
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1005;
        Hf          0;
    }

    transport
    {
        As   1.458e-6;
        Ts   110.4;
    }
}
```

```
47
48 // ************************************************************//
```

4.2.3 Selection of the discretization schemes

Because the flow is near sonic conditions, it has been decided to use for the divergence terms, differentiation schemes towards the upwind in order to guarantee the boundedness of the solution. In particular for the velocity and the fluxes on the CV faces it has been used the *GaussLimtedLinear* differentiation scheme (see Table 3.3), while to guarantee the complete stability of the turbulence a pure upwind scheme has been used for ω and k. To proceed forward in time the *Euler* scheme has been selected, this requires to achieve temporal accuracy and numerical stability the constraint of a Courant number less than 1. In the monodimensioanl case the Courant number is defined as:

$$Co = \frac{\mathbf{u}\Delta t}{\Delta x} \tag{4.24}$$

where Δt is the time step of each iteration and Δx the cell size. If it is smaller than 1, it guarantee that the information at a certain time step t^n comes from the prevoius time step t^{n-1} and from the neighbor cell. For all the simulation in these thesis the maximum Courant number has been set to 0.5 in the *controlDict* file. For the Laplacian terms the *Gauss linear corrected* scheme has been selected and to evaluate it a linear interpolation scheme between the CV faces has been chosen. To account the presence of non-orthogonalities in the mesh especially in the transitions between blocks 1-2 and blocks 3-4 the corrected option has been selected for the *snGradSchemes*. Finally, to evaluate the distance from the wall in the wall-functions of the $k-\omega$ SST model the *meshWave* method has been defined.

```
1  /*--------------------------*- C++ -*----------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD
4     \\    /   O peration     | Website:  https://openfoam.org
5      \\  /    A nd           | Version:  7
6       \\/     M anipulation  |
7  \*---------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSchemes;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 ddtSchemes
19 {
20     default         Euler;
21 }
22
23 gradSchemes
```

```
24 {
25     default          Gauss linear;
26 }
27
28 divSchemes
29 {
30     default          none;
31     div(phi,U)       Gauss limitedLinearV 1;
32     div(phi,e)       Gauss limitedLinear 1;
33     div(phid,p)      Gauss limitedLinear 1;
34     div(phi,K)       Gauss limitedLinear 1;
35     div(phiv,p)      Gauss limitedLinear 1;
36     div(phi,k)       Gauss upwind;
37     div(phi,omega)   Gauss upwind;
38     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
39 }
40
41 laplacianSchemes
42 {
43     default          Gauss linear corrected;
44 }
45
46 interpolationSchemes
47 {
48     default          linear;
49 }
50
51 snGradSchemes
52 {
53     default          corrected;
54 }
55
56 wallDist
57 {
58     method meshWave;
59 }
60
61 // ************************************************************//
```

4.2.4 Selection of the equation solvers

For all the simulation variables the selected solver is the *smoothSolver* with the Gauss-Seidel method as a smoother. The tolerance to be reached for each variable before the solver stops is defined by the keyword *tolerance*.The only exception is for the fluid's density in the continuity equation, which is calculated using the diagonal solver that as described is section 3.3.2 is a directed method. This allows to have a density value free of error in order to avoid numerical instabilities due to the near sonic conditions. The PIMPLE algorithm has been set with two outer loops and just one inner loop, these parameters are the most used inside the OpenFOAM's tutorials and they are suggested in the OpenFOAM's user guide. Because using

the utility *checkMesh* the maximum mesh non-orthogonality has a value of 30.86 and because the *corrected* option has been used for the evaluation of the surface normal gradient, a non-orthogonal corrector has been added. Finally, because the flow is near sonic condition the transonic option has been switched to yes. With this option active, the pressure and velocity correction inside the PIMPLE algorithm are strongly relaxed to avoid the blow-up of the solution.

```
/*---------------------------*- C++ -*----------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
{
    "rho.*"
    {
        solver          diagonal;
    }

    "p.*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }

    "(U|e|R).*"
    {
        $p;
        tolerance       1e-05;
    }

    "(k|omega).*"
    {
        $p;
        tolerance       1e-08;
    }
}

PIMPLE
```

```
{
    nOuterCorrectors 2;
    nCorrectors      1;
    nNonOrthogonalCorrectors 1;

    transonic            yes;
}

// ************************************************************//
```

4.2.5 Boundary and initial conditions

The boundary conditions are defined in the 0 directory and they are imposed for all the simulation's variables (U, p, T, ω, k, ν_T and α_T). Below is reported a scheme with the names assigned to each patch of the domain:

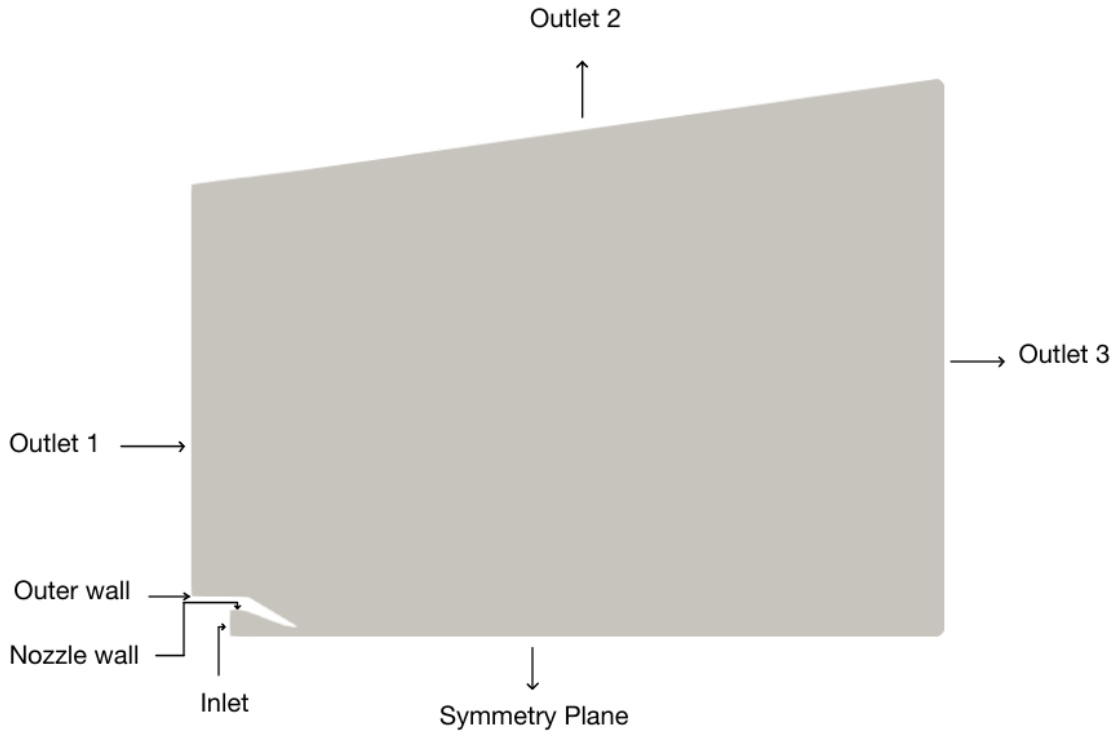


Figure 4.5: Near sonic jet case OpenFOAM's patches

For the inlet the total pressure (1.861 bar) and the total temperature (294.4 K) have been defined as initial condition for the pressure and the temperature, while for the velocity has been imposed a zero gradient condition. The outlets have been join together as an only patch with the name of freestream and for them the *wave-Trasmissive* boundary condition have been applied for both pressure and velocity. This boundary condition is specific for high speed flows and it avoids spurious wave reflections that would be detrimental for the simulation. The boundary condition for the free-stream temperature has been set to zero gradient. For the nozzle wall and the outer wall a noSlip condition have been imposed for the velocity and a

zeroGradient condition have been imposed for both temperature and pressure. In the internal part of the domain according to the simulation performed by NASA the temperature has been set to a value of 294.4 K (the same value of the inlet total temperature), the pressure to the value of the atmospheric pressure (1 bar) and for the velocity a very low background ambient condition ($M_{ref} = 0.01$ corresponding to approximately 3.54 m/s) has been imposed. This is necessary because flow into quiescent air is very difficult to achieve for most CFD codes. The turbulent quantities k and ω need an initial value for the start of the simulation, this can be approximated using the relations for isotropic and homogeneous turbulence. Supposing the velocity fluctuations u'_x, u'_y and u'_z equal to 5% of the nozzle exiting velocity (approximately 310.46 m/s) and a length scale l equal to the nozzle diameter $D = 0.0508$ m

$$k = \frac{1}{2}\left(u_x'^2 + u_y'^2 + u_z'^2\right) = \frac{3}{2}\left(0.05 \cdot U_{out}\right)^2 = 361.32\frac{m^2}{s^2} \tag{4.25}$$

$$\omega = \frac{C_\mu^{0.75} k^{0.5}}{D} = 61.48 s^{-1} \tag{4.26}$$

These values are set as initial values for all the internal part of the domain as well as for the outlets and the inlet patches. Because the average $y+$ on the nozzle's wall has a value of 0.682 no-wall functions are used on this patch, on the duct wall instead, the average $y+$ has a value of 17.42 therefore the wall functions of the $k-\omega$ SST model have been used. Finally, regarding α_T it is the turbulent thermal diffusivity defined as $\alpha_T = \frac{\mu_T}{Pr_T}$ with Pr_T the turbulent Prandtl number considered constant with a value of 0.9. This term arise after the Favre-Averaging of the energy equation and it induce an additional thermal diffusivity in the turbulent boundary layer that enhances the heat transfer due to convection. For this last quantity the value has been set to zero on all the patches and in the internal mesh with the only exceptions of the duct wall where the *alphatWallFunction* has been used. The complete files with the here described boundary conditions can be found in Appendix B.

4.2.6 Simulation control

The parameters for the control of the simulation are specified in the *controlDict* file. Since no residual controls have been specified for the simulation variables, the simulation time has been set to 0.4 s. This correspond approximately to 13 flow through periods with the conservative assumption of a jet average velocity of 83 m/s. As stated in section 4.2.3 the maximum Courant number has been set to 0.5, therefore the *runTimeModifiable* and *adjustTimeStep* have been switched to yes. To check the near sonic conditions at the nozzle exit the function *MachNo* has been used to evaluate the Mach number at each solver iteration.

```
1 /*---------------------------*- C++ -*----------------------*\
2   =========                 |
3   \\      /  F ield         | OpenFOAM: The Open Source CFD
4    \\    /   O peration     | Website:  https://openfoam.org
5     \\  /    A nd           | Version:  7
6      \\/     M anipulation  |
7 \*-------------------------------------------------------*/
8 FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoPimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         0.4;

deltaT          1e-6;

writeControl    adjustableRunTime;

writeInterval   0.01;

purgeWrite      2;

writeFormat     ascii;

writePrecision   8;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep    true;

maxCo 0.5;

maxDeltaT 1e-2;

functions
{
    #includeFunc MachNo
}

// ************************************************************//
```

4.3 Results and comparisons

In this section are reported the simulations results. They are compared to the ones provided by the NASA validation case and the experimental results of [31] and [32].

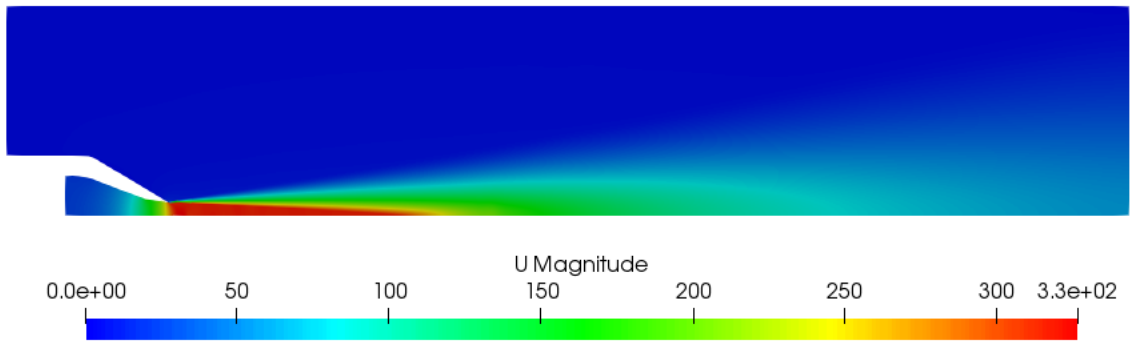


Figure 4.6: U magnitude contour plot

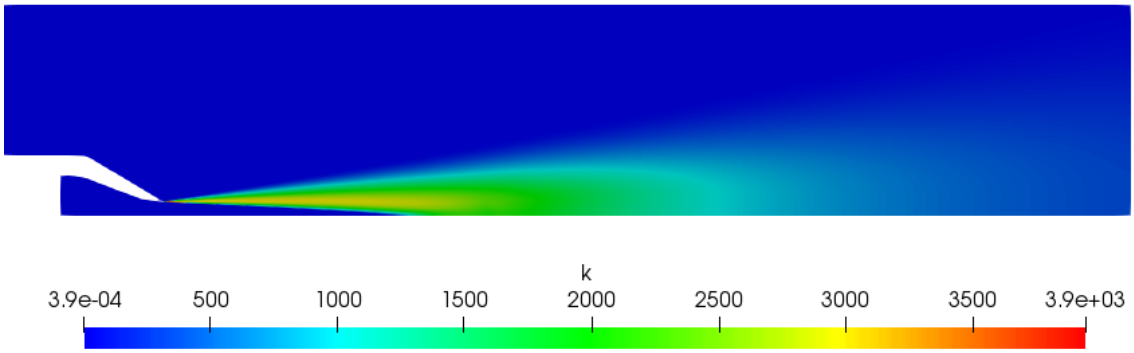


Figure 4.7: k contour plot

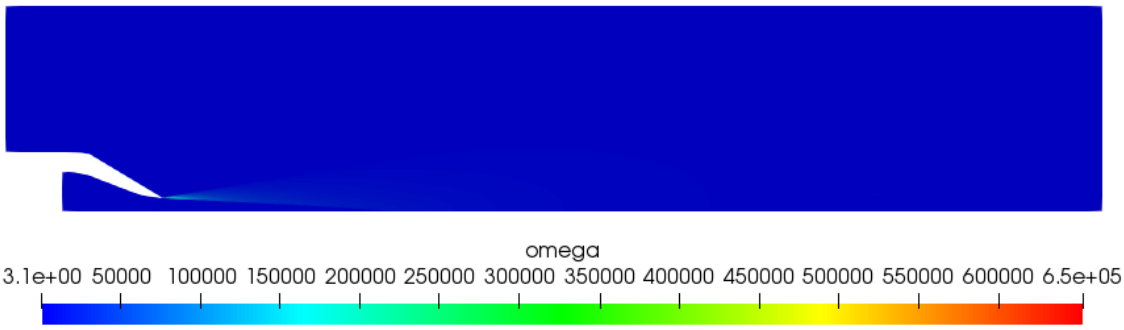
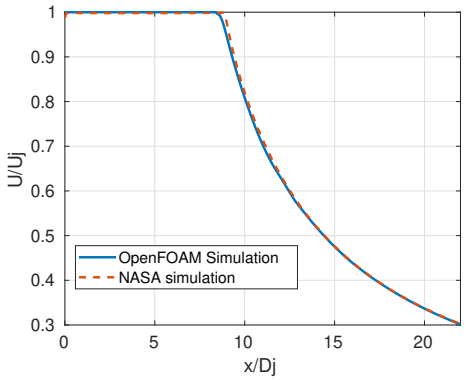
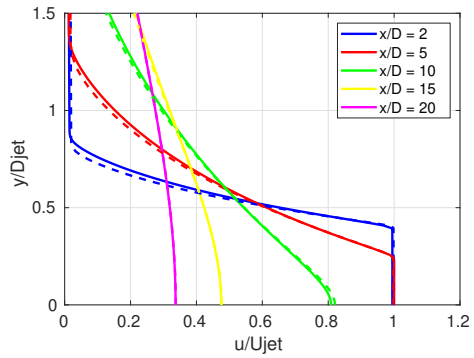


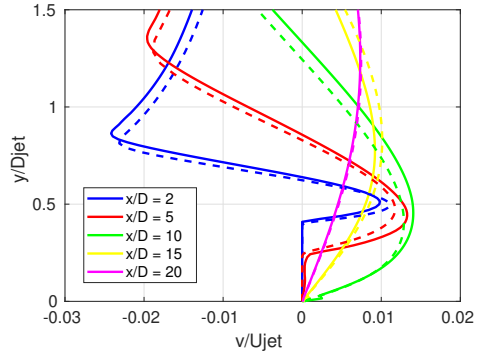
Figure 4.8: ω contour plot



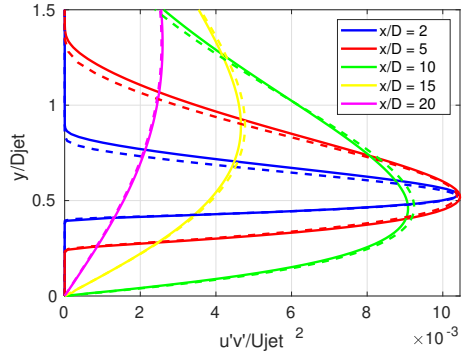
(a) Centerline velocity



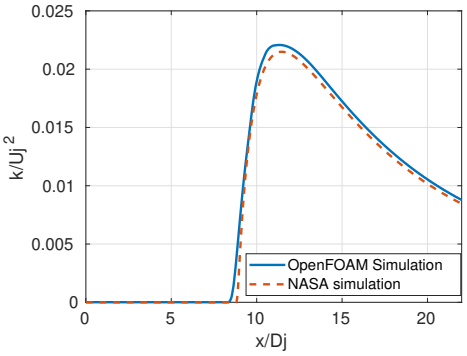
(b) u_x profiles



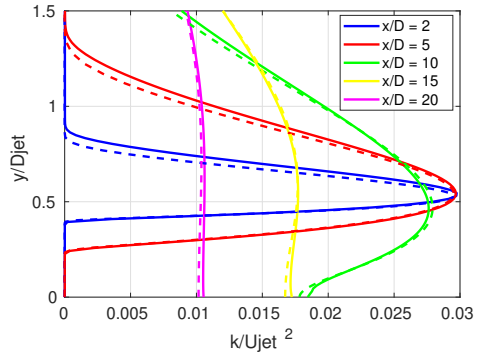
(c) u_y profiles



(d) $u'v'$ profiles

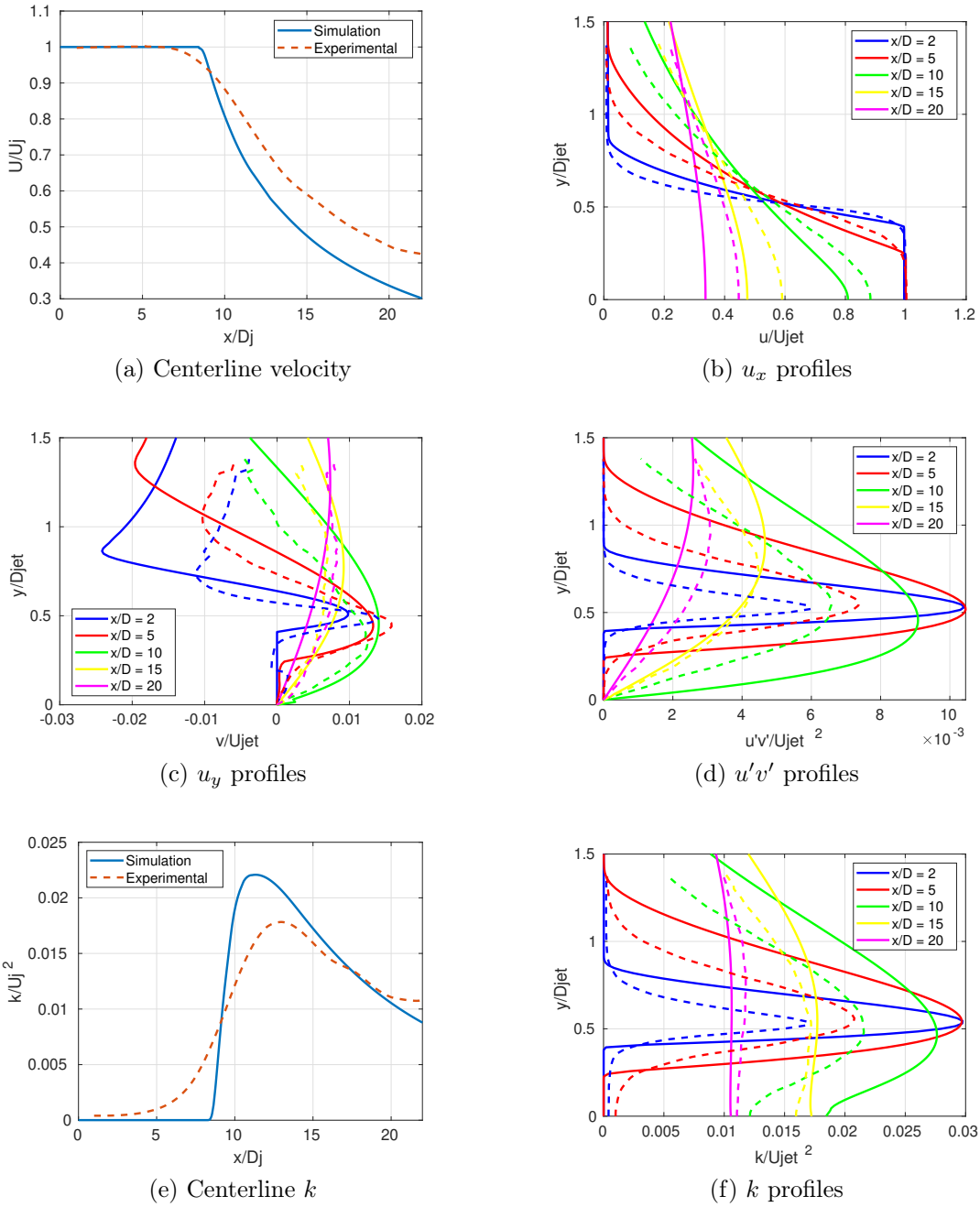


(e) Centerline k



(f) k profiles

Figure 4.9: Simulations comparisons OpenFOAM (solid line -), NASA (dashed line - -)



(a) Centerline velocity

(b) u_x profiles

(c) u_y profiles

(d) $u'v'$ profiles

(e) Centerline k

(f) k profiles

Figure 4.10: Simulations comparisons OpenFOAM (solid line -), Experimental data (dashed line - -)

Looking at Figure 4.9 it is clear how the NASA simulation data and the ones obtained using OpenFOAM are in very good agreement. This is particularly true if it is considered that the two simulations are performed on two different grids and with a small difference in the production term of the $k - \omega$ SST turbulence model. The experimental results of Figure 5.15 show instead how the simulation overpredicts the turbulence, this results in a higher turbulent kinetic energy with the consequent increase of the turbulent viscosity. With an higher turbulent viscosity, the jet becomes over-diffusive and this can be seen in the spreading rate of the centerline velocity which is faster after the end of the jet's potential core. However, looking at figure 4.11, it is evident that approaching the self-similar region ($x/d >$

30) the decay of the centerline velocity is very similar to the analytical one, provided by the relation:

$$\frac{U_0(x)}{U_j} = \frac{B}{(x - x_0)/d} \tag{4.27}$$

Where d is the nozzle diameter, U_j is the velocity at the nozzle exit, x_0 is the jet virtual origin and B is the velocity decay constant, with a value of approximately 6. This relation has no dependence on the Re number and it can be analytically demonstrated using the self-similarity hypothesis, based on the experimental observations that, as the jet decays and spread, the mean velocity profiles changes but the shapes of the profiles does not change. The interested reader can find a complete discussion about turbulent round jets and free shear flows in [34]

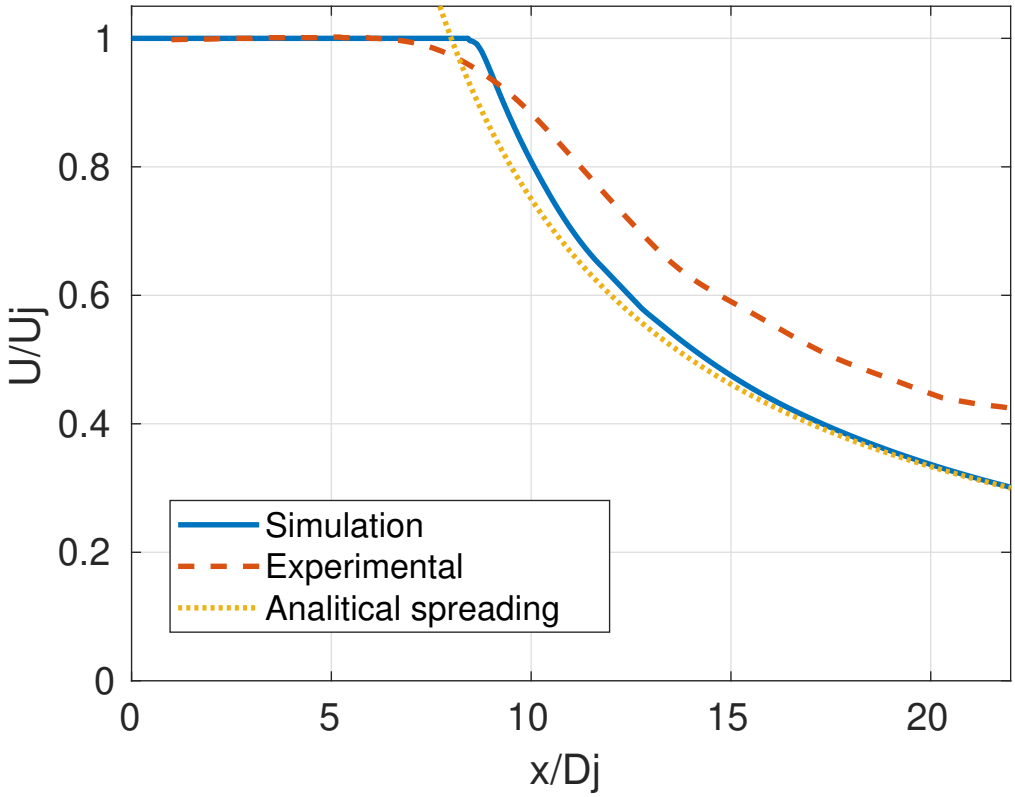


Figure 4.11: Analytical spreading rate

As far as has been said, it is possible to say that with this setting the solver rhoPimpleFoam is well able to predict sonic jet flows. It provides results that are in very good agreement with the NASA near sonic jet validation case and it is able to correctly predict the analytical jet spreading given by (4.27). The pour agreement with the experimental results is mainly due to the RANS approach and because for axisymmetric cases like this, it is important to solve the turbulence dissipation terms in a strong conservative form.

Chapter 5

Axisymmetric simulations in flight condition

In this chapter are reported the results for the axisymmetric simulations, with periodic-rotating boundary conditions, of the coflow jet exiting from a CFM-56 engine. These simulations have as their main aim the modeling of the first phase in the contrail formation,the *Jet Regime*. They are developed only to account the fluid-dynamics of the phenomenon, which means that no soot particles are considered inside the flow. This strongly simplifies the calculations, avoiding to use a Lagrangian approach that takes in consideration the effect of the particles on the fluid. In the Jet Regime the jet is expanded into the atmosphere and mixed with the ambient air, during this phase one assumes that the jet expansion is not influenced by the aircraft vortex formation and that it covers a distance of approximately 30 m.[35]. After the definition of the flight conditions, the two types of performed simulations are presented, in the first one is not considered the wall effect of the duct surrounding the engine while in the second one this effect is taken in consideration. The results of the two simulation are finally compared and discussed.

5.1 Flight conditions

The CFM-56 engine is a two-flux turbofan engine where the exit of the effluents (core flow) is surrounded with a cold air flow (bypass flow) and it is mainly used on Boeing 737 aircraft. In the simulation the effect of the bypass flow has not been considered and the geometry of the drain nozzle has been extremely simplified. In flight cruising condition, this engine can develop around $32,900$ pounds of thrust ($7,393\ N$) with the exhaust gases exiting at 480 m/s with a temperature of 580 K. Table 5.1 taken from [36], reports the internal characteristics of the engine, they will be used in the definition of the simulation's initial conditions.(*Note* : In Table 5.1 u_s, u_p and c_s, c_p are respectively, the velocity and the spreed of sound of the bypass and core streams.)

5.2 Axisymmetric no-wall case

The first step of the simulation has been the definition of a suitable geometry. The downstream domain dimension from the nozzle exit has been set to 30 m in order

to simulate the total length of the jet regime.

CFM-56 Engine	
Core flow: Mach	1
Static pressure P (Pa)	24,000
Static Temperature T (K)	3
u_p (m/s)	480.3
c_p (m/s)	480.3
Flow area S (m^2)	0.292
AFR (air fuel ratio)	60/1
Mass flow (kg/s)	20.2
Bypass flow: Mach	1
Static pressure P (Pa)	31,700
Static temperature T (K)	242
u_s (m/s)	311.6
c_s (m/s)	311.6
Flow area S (m^2)	0.749
Mass flow (kg/s)	106.5
Flight conditions: Mach	0.85
Ambient pressure Ps (Pa)	23,800
Ambient temperature T (K)	219
u (m/s)	252
c (m/s)	296.6

Table 5.1: Internal engine characteristics

Starting from the nozzle exit the vertical dimension is set to 10 m and it proceeds forward with an inclination of 7.4°. According to the geometrical characteristic of the CFM-56 drain nozzle, the inlet and outlet nozzle's radius have a values of 0.305 and 0.915 m respectively. Figure 5.1 shows the geometry dimensions as a function of the nozzle exit radius.

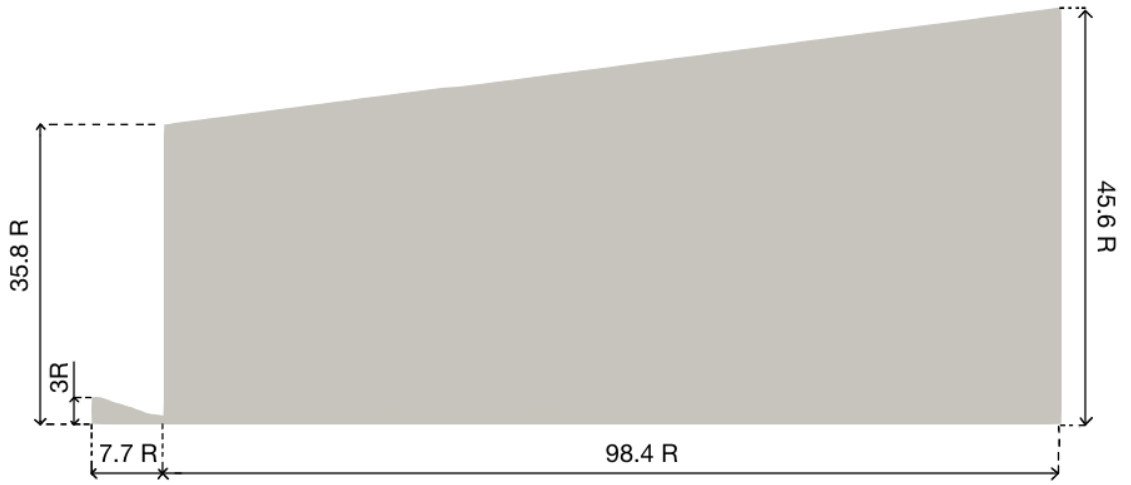


Figure 5.1: No wall case geometry dimensions

5.2.1 Mesh generation

As for the NASA near sonic validation case, a wedge shape mesh with periodic rotating boundary conditions has been created using the utility *BlockMesh*. In this case the domain has been divided in 3 blocks and it has been tried to put the majority of the cells along the downstream direction of the nozzle exit and in the 2 first diameters over the jet centerline. Indeed, the high value of the coflow velocity, strongly limits the jet expansion, allowing the flow's stronger gradients to be inside this region. To guarantee a smooth transition between each block, the dimension of the cells obey to a geometrical progression law, which has as starting point the nozzle exit. Along the radius and the axis of symmetry of the nozzle, the cells have no expansions ratio, then continuing along the axial direction, the cells slowly increase their length, while their height along the y-direction remains constant until the nozzle radius. Over the nozzle radius the height of the cells starts to increase until reaching the wanted expansion ratio at the top of the domain. Since the starting point in the expansion of the cells is the upper boundary of the nozzle wall (better known as nozzle lipline) and the number of cells in the nozzle is the parameter through which starts the building of the mesh, the correct number of cells in each block can be evaluated following the here reported procedure:

- Denoting with $n_{x,noz}$ and $n_{y,noz}$ the number of cells in the x and y direction of the nozzle and considering no expansion ratio inside the nozzle's block, the cell lengths along x and y at the nozzle lipline, can be evaluated respectively as:

$$\Delta_x = \frac{l_{noz}}{n_{x,noz}} \text{ and } \Delta_y = \frac{r_{noz}}{n_{y,noz}} \tag{5.1}$$

- Considering that the cell elements along the edges of a block are in geometrical progression, it is possible to write:

$$l_{edge} = \Delta_0 \frac{q^n - 1}{q - 1} = \Delta_0 \frac{Ex.ratio^{\frac{n}{n-1}} - 1}{Ex.ratio^{\frac{1}{n-1}} - 1} \tag{5.2}$$

  Where n is the number of cells along the block's edge and depending on the edge orientation; Δ_0 is the length along x or y of the first cell of the edge. This length is imposed to be equal to the corresponding nozzle's cell. In this way it is guaranteed that the cells at the end of the nozzle's block and at the beginning of the new block have the same dimensions.

- Because Δ_0 as well as the length of the block's edge l_{edge} are known parameters (5.2) can be resolved as a function of n, allowing to obtain the number of cells of a block's edge for a fixed expansion ratio. This procedure can also be repeated for blocks that are not bordering with the nozzle block and allows to build an hexahedral structured mesh.

To perform a grid independence analysis three grids with increasing refinement have been generated and below are reported a sketch with the blocks distribution (Figure 5.2), the Tables with the specifications for each mesh (5.2 - 5.4) and the front section of the most refined mesh (Figure 5.3).

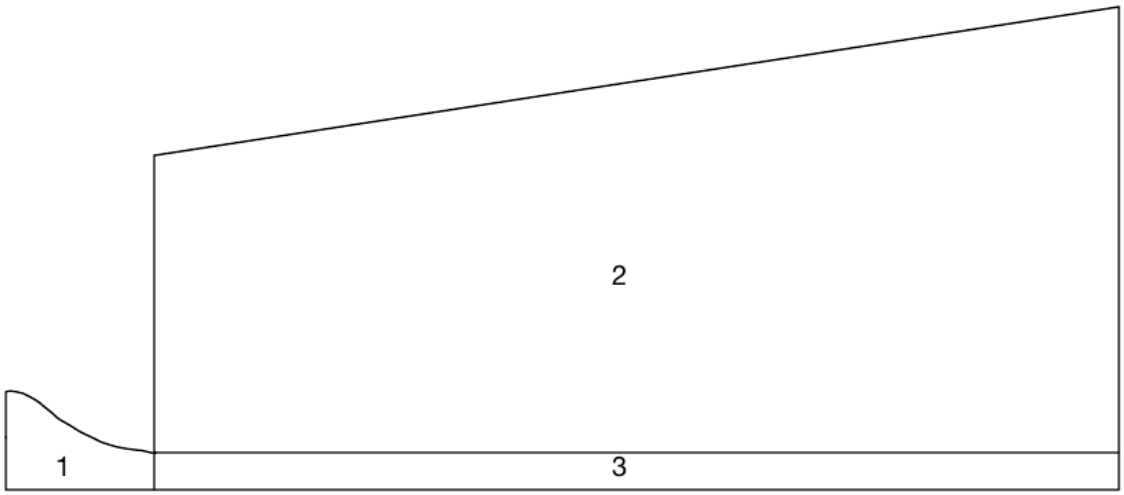


Figure 5.2: Blocks in the axisymmetric no-wall case (Figure not in scale)

	Mesh 1 63,194 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	80	70	1	1	1
Block 2	331	104	1	7	100
Block 3	331	70	1	7	1

Table 5.2: Mesh 1 axisymmetric no-wall case

	Mesh 2 139,927 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	110	100	1	1	1
Block 2	563	129	1	5	120
Block 3	563	100	1	5	1

Table 5.3: Mesh 2 axisymmetric no-wall case

	Mesh 3 202,462 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	150	120	1	1	1
Block 2	619	178	1	5	100
Block 3	619	120	1	5	1

Table 5.4: Mesh 3 axisymmetric no-wall case

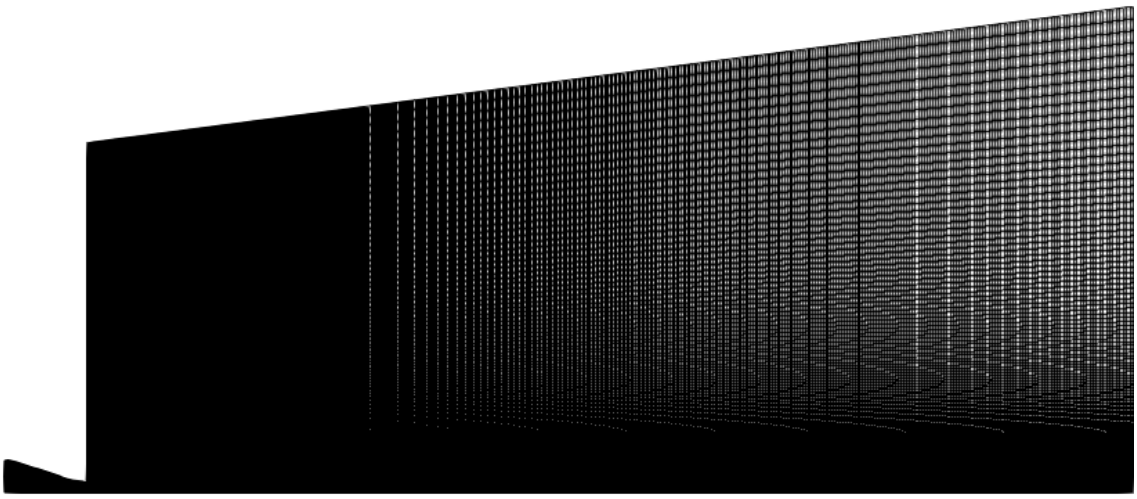


Figure 5.3: Mesh 3 axisymmetric no-wall case front section

5.2.2 Boundary and initial conditions

In this new case two inlet boundary conditions have been inserted to account the presence of the coflow due to the aircraft traveling at flight cruise condition (5.4).The first one at the nozzle inlet to account the main flow and the second one at the beginning of block 2 to account the coflow. As for the NASA near sonic validation case, the two outlets have a waveTransmissive boundary condition for both the pressure and the velocity while the nozzle wall has noSlip boundary condition for the velocity.

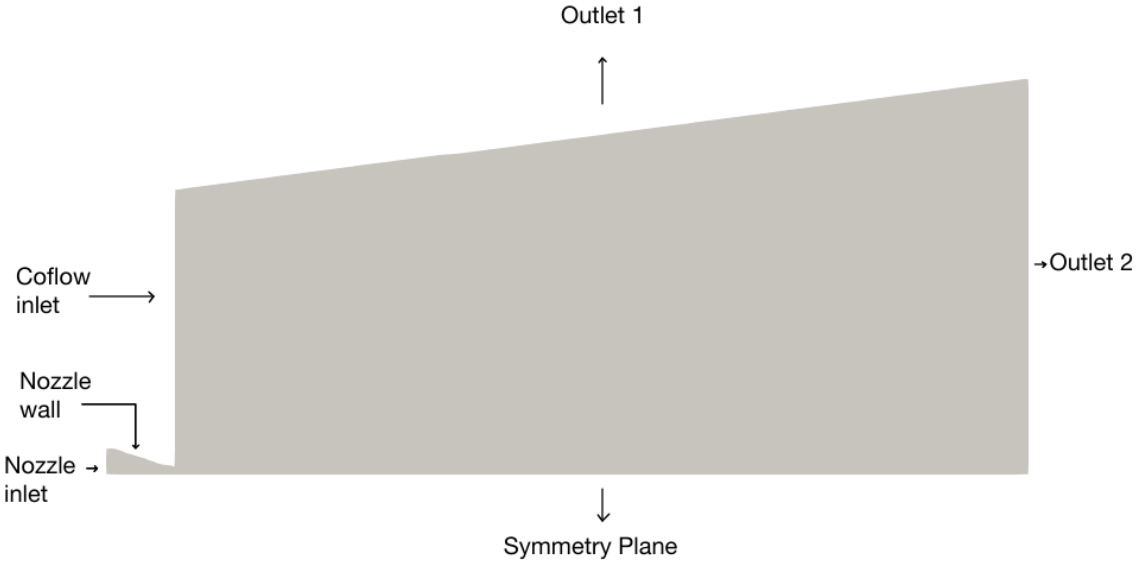


Figure 5.4: No wall case boundary conditions

At the nozzle inlet, total pressure and temperature boundary conditions have been imposed. These can be calculated using the values of temperature,pressure and Mach number at the nozzle exit reported in Table 5.1. With the hypothesis of

an *isentropic* expansion inside the nozzle it is possible to write:

$$p_{tot} = p_{out}\left[1 + \frac{1}{2}(k-1)\,Ma_{out}^2\right]^{\frac{k}{k-1}} = 45430.3Pa \tag{5.3}$$

$$T_{tot} = T_{out}\left(1 + \frac{k-1}{2}Ma_{out}^2\right) = 696K \tag{5.4}$$

For the coflow inlet fixed values for temperature and velocity have been imposed. These values are the same of the ambient condition reported in Table 5.1. The pressure instead, has a zero gradient boundary condition on this patch. The ambient conditions of temperature, pressure and velocity have also been imposed as initial values in the internal part of the mesh. For the turbulent quantities k and ω the same hypothesis of Section 4.2.5 have been made. However, because the two values of velocity at the two inlets are different, there are two initial condition for each of the turbulent quantities. Using equations (4.25) - (4.26) and the velocity values at the nozzle exit and in the ambient coflow, the initial conditions for the turbulent quantities at the two inlets are reported in the following table:

	Nozzle inlet	**Coflow inlet**
k_0 $[m^2/s^2]$	864	238.14
ω_0 $[s^{-1}]$	7.92	4.16

Table 5.5: Turbulent quantities initial conditions

Because in this case the fluid reaches higher velocities during its expansions in the nozzle, the $y+$ is higher than 10 on all the three generated meshes, this requires the use of the wall functions on this patch to have a good wall treatment. Finally, the parameter α_T has been set to zero in all the domain. The boundary and initial conditions together with the *BlockMeshDict* file for the axisymmetric no wall case can be found in Appendix C.

5.2.3 Simulation control

The simulation set up is the same described in Chapter 4, however due to the larger domain and the high number of cells the simulation has been run in parallel using ten processor. This is very easy in OpenFOAM, that thanks to the *decomposePar* utility and the *scotch* decomposition method can decompose the simulation domain in different part of the same size and assign each of them to a processor. The decomposed simulation has been run on Dragon the cluster of the Illinois University of Chicago (UIC). Dragon has 18 nodes with 18 cores and 64GB or RAM for each of them. Each processor belongs to Intel Xenon family with a maximum speed of 4GHz. Further informations on the Dragon cluster can be found at [37]. To guarantee the reaching of the steady state, the simulation time has been set to 1.5 s, that with the conservative assumption of an average flow velocity equal to the one of the coflow, corresponds approximately to 13 flow through periods. Again the Courant number has been fixed to 0.5 allowing the time step to be variable. Because the Courant number is fixed and the time-step variable, it has been decided to not perform a time convergence study.

5.2.4 Results and validation

To check the validation of the results the the simulation centerline velocity and the simulation potential core length have been compared to the analytical relation available in [38].

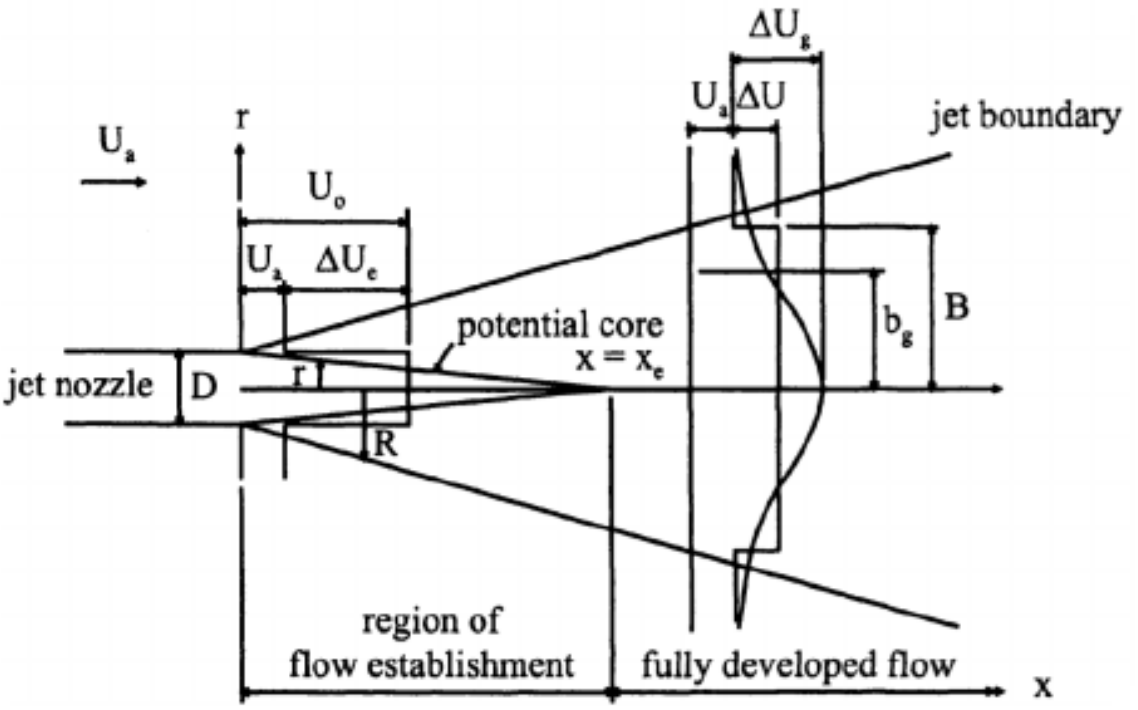


Figure 5.5: Diagram of round jet in coflow

Looking at Figure 5.5 the relevant physical quantities of a turbulent jet in coflow are the velocity exiting from the nozzle U_o, the coflow velocity U_a, the potential core length x_e, the half-width b_g and the top half-width $B_g = \sqrt{2}b_g$.
The half-width b_g is defined as the height of the velocity profiles in which the axial component of the velocity has a value which is $1/e$ of the centerline jet velocity U_g. It is assumed to spread constant $|u_e|\frac{db_g}{dx} = \beta|u_e|$, with the excess velocity u_e expressed as :

$$u_e = \begin{cases} \Delta U \text{ if } r \leq B \\ 0 \text{ otherwise} \end{cases} \tag{5.5}$$

This is equivalent to a jet with a sharp boundary and uniform velocity $\Delta U + U_a$, carrying the same mass flow and excess momentum of the actual jet. With these hypothesis and remembering that in a jet in coflow the excess momentum $M_{eo} = (U_o - U_a)U_oA_o$ is conserved, it is proved that $\Delta U = \frac{\Delta U_g}{2} = \frac{U_g - U_a}{2}$ while the top-half width B and the excess velocity ΔU are related through the system of equations:

$$\begin{cases} U^{*2} + U^* - \frac{1}{\pi B^{*2}} = 0 \\ \frac{dB^*}{dx} = \beta_s \frac{U^*}{1+U^*} \end{cases} \tag{5.6}$$

Where $U^* = \Delta U/U_a$, $B^* = B/l_m^*$, and $x^* = x/l_m^*$ are dimensionless variables, $l_m^* = M_{eo}^{1/2}/U_a$ is the excess momentum length scale and β_s is a model constant equal to 0.16. Equation (5.6) can be numerically integrated to obtain a solution for

$\Delta U(x)$ and $B(x)$, and consequently for the jet centerline velocity $U_g(x)$.
Adopting the same hypothesis the length of the potential core x_e can be estimated with the relation:

$$\frac{x_e}{D} = \frac{\sqrt{1 + U_a/U_o}}{\beta_s(1 - U_a/U_o)} \tag{5.7}$$

Equation (5.6) needs initial conditions relative to the jet virtual origin to be integrated:

$$\begin{cases} \Delta U_o^* = \frac{\Delta U_o}{U_a} \\ B_o^* = \frac{D}{2l_m^*} \\ x_o^* = \frac{-D}{2\beta_s(1-U_a/U_o)} \end{cases} \tag{5.8}$$

This is applicable only in the region after the jet flow is fully developed ($x > x_e$). To have an idea of the full evolution of the jet centerline velocity, it is possible to impose for $x \leq x_e$ the centerline velocity equal to U_o, then after determining the length of the potential core x_e, the integration of the governing equations can start from $x_o^* = x_e/l_m^*$. In a more simple way it can also be demonstrated that in the near field ($x/l_m^* \leq 10$) the centerline excess velocity scale as x^{-1} (5.9) while in the far field ($x/l_m^* \geq 60$) as $x^{-2/3}$ (5.10).

$$\frac{\Delta U_g}{U_a} = 7.0 \left(\frac{x}{l_m^*}\right)^{-1} \tag{5.9}$$

$$\frac{\Delta U_g}{U_a} = 2.14 \left(\frac{x}{l_m^*}\right)^{-\frac{2}{3}} \tag{5.10}$$

After this brief explanation of what to expect for a jet in coflow here are reported the contour plots for the main simulation variables:

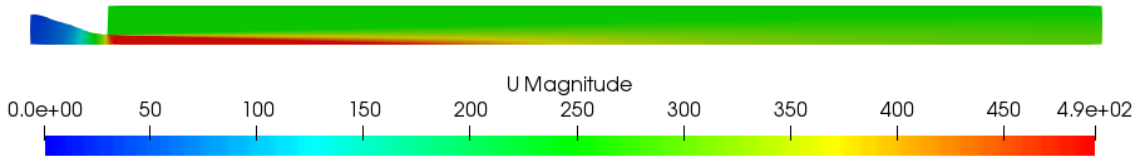


Figure 5.6: U magnitude contour plot no wall case

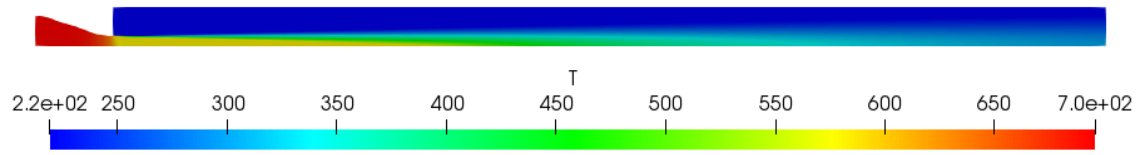


Figure 5.7: T contour plot no wall case

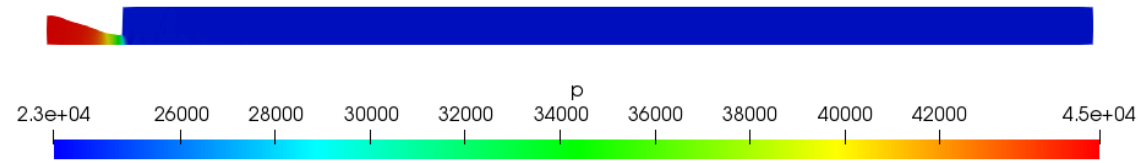


Figure 5.8: p contour plot no wall case

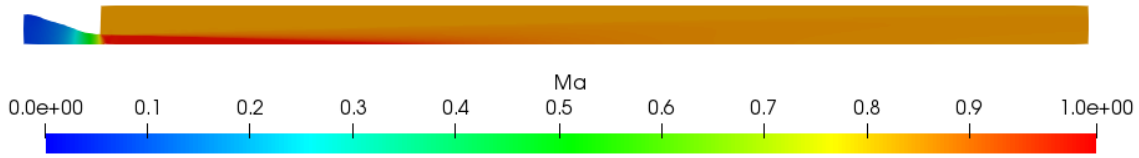


Figure 5.9: Ma contour plot no wall case

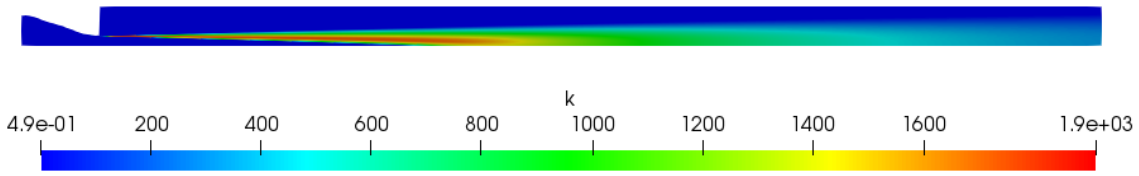


Figure 5.10: k contour plot no wall case

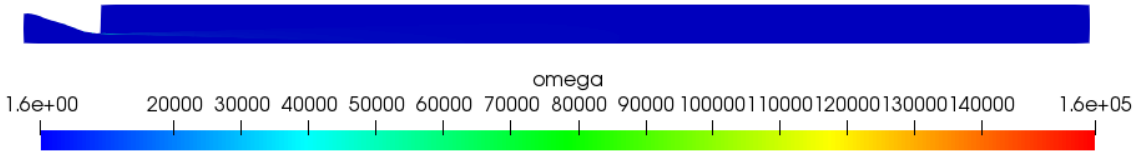


Figure 5.11: ω contour plot no wall case

Figures 5.6 - 5.11 are obtained on Mesh 2. It is clearly visible how the ambient coflow, strongly limits the jet expansion allowing the jet potential core to be longer. It is also possible to observe how the the initial conditions at the jet inlet, imposed using (5.3) and (5.4) led to values of Mach number, temperature and velocity that are in agreement with the ones specified in Table 5.1. With these boundary condition the jet has a centerline nozzle exit velocity of 484.4 m/s and an average Re number of $1.36 \cdot 10^6$. Looking at the contour plots of the turbulent kinetic energy (Figure 5.10) and turbulent dissipation rate (Figure 5.11) it may be noticed how the nozzle lipline is the critical zone regarding the turbulence quantities. It is along this line that the coflow starts mixing with the flow exiting from the nozzle, expanding gradually the thickness of the boundary layer which reaches the axis of the nozzle at the end of the potential core, where the turbulent kinetic energy has its peak. Inside this region the smaller turbulent eddies begin to develop subtracting energy from the flow exiting from the nozzle and allowing the jet velocity profiles to become self-similar.
Here are now reported the evolution of the U_x, T, and k along the centerline. The quantity U_x is in particular compared with the analytical solution given by (5.6).

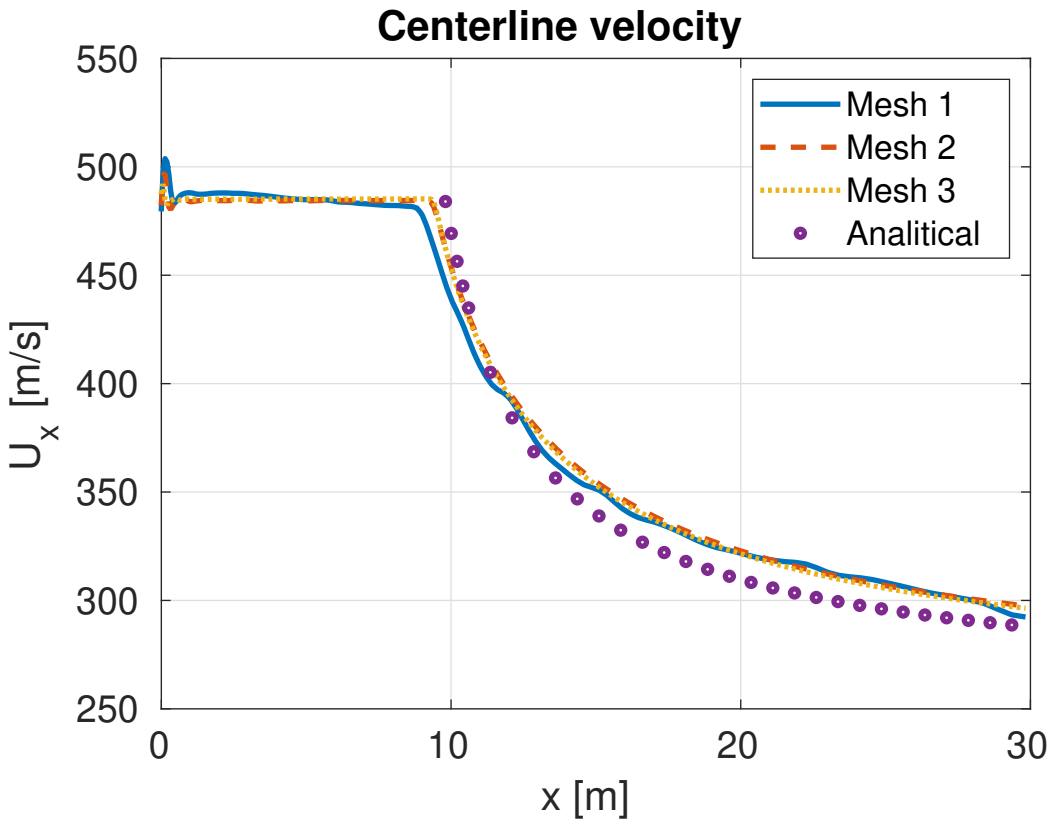


Figure 5.12: No wall case centerline velocity

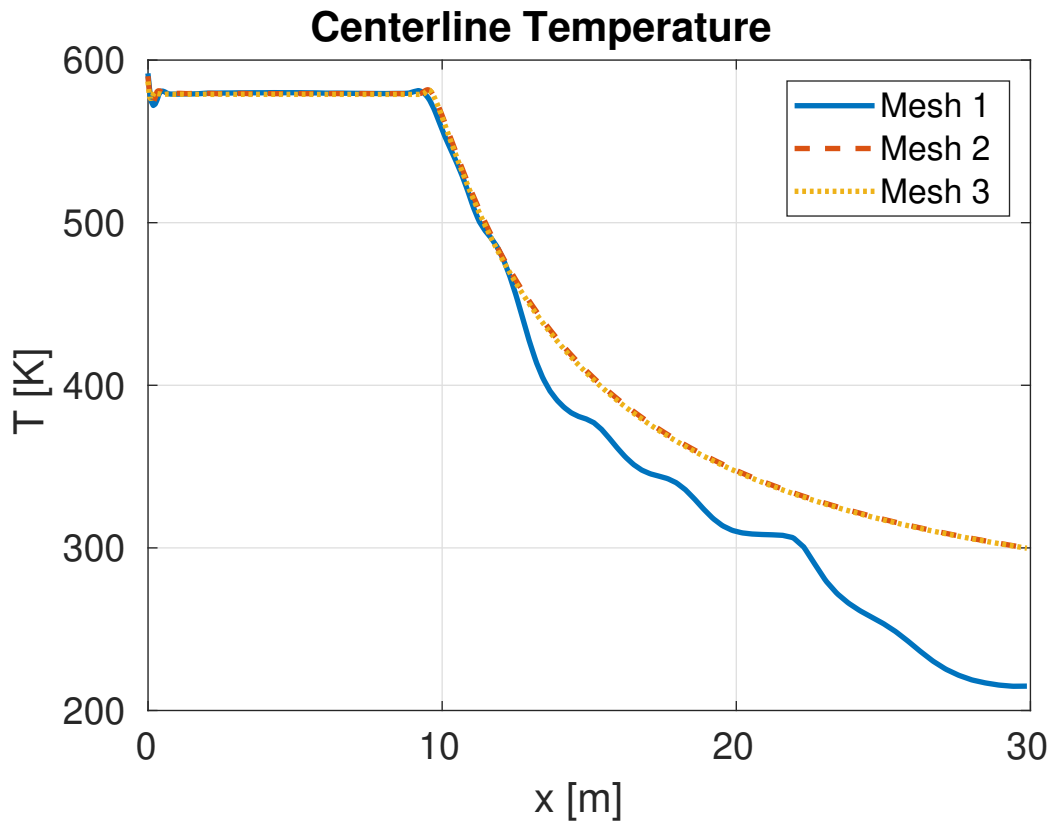


Figure 5.13: No wall case centerline temperature

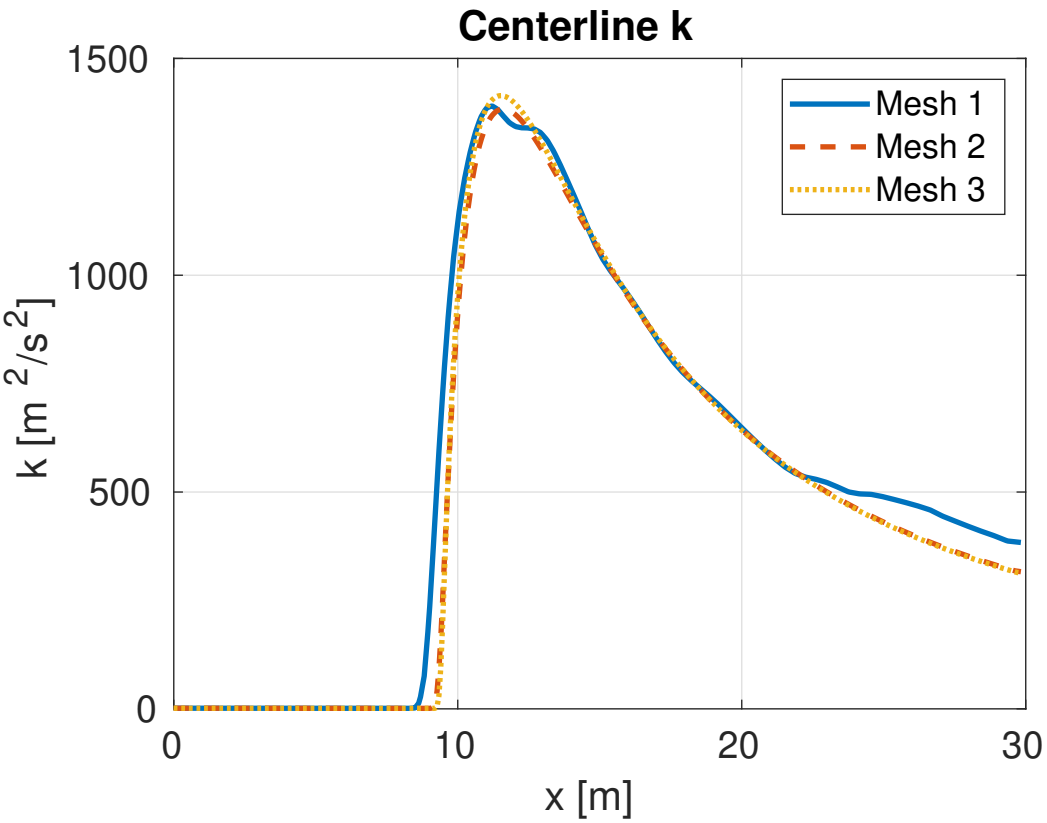
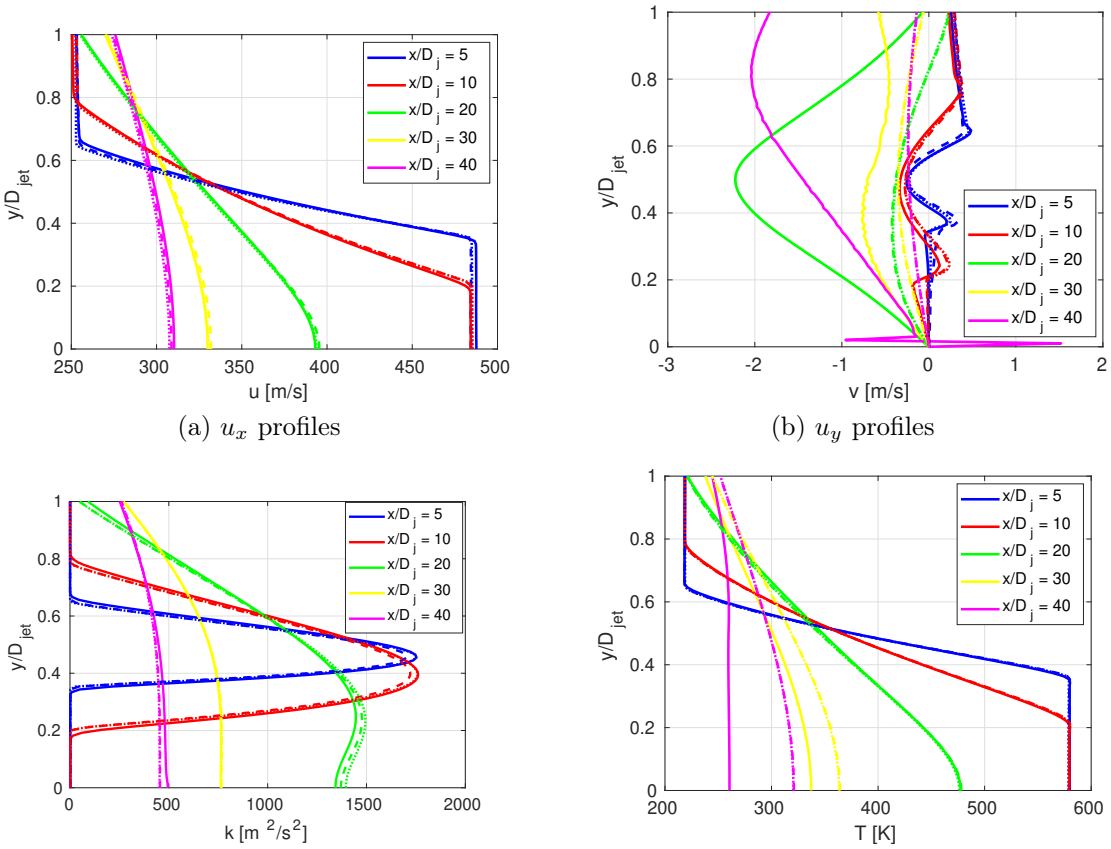


Figure 5.14: No wall case centerline k

Looking at Figure 5.12 it is evident that all the simulations predict a lower decay of the centerline velocity compared to the analytical solution given by (5.6). The analytical solution is obtained using the initial conditions provided by (5.8), they are computed using a value of U_o equal to the one provided in Table 5.1. This velocity value is also used to estimate the potential core length with (5.7), in order to obtain a starting value $x_o^* = x_e/l_m^*$ for the integration of the equations. All the three figures show how the first grid is not enough refined to guarantee a converged solution, indeed for $x > 10\ m$, U_x, T, and k have fluctuations denoting the not sufficient mesh refinement over this distance. The temperature in particular, has a sharp drop with a value of $215K$ at the end of the domain which is completely an-physical keeping in mind that the ambient temperature has a value of $219K$. Mesh 2 and Mesh 3 have results that are in very good agreement one to each other, the potential core length is respectively of 9.377 m and 9.422 m with a relative error of 5.32 % and 4.87 % respect to the analytical value (9.904 m). This value is also in agreement to the one provided by [36] as well as the temperature decay along the nozzle downstream direction. Figure 5.15 shows the profiles of u_x, u_y, k and T along the downstream direction of the nozzle. For all the profiles it is evident how they become self-similar after the potential core. The only exception is the velocity vertical component and temperature profiles obtained with the first mesh, they are far from the ones obtained with Mesh 2 and Mesh 3 showing again the not sufficient grid refinement for $x > 10\ m$. Looking at the turbulent kinetic energy profile, it is interesting to notice how proceeding along x, the profile peak gradually goes down, it starts from the lipline $y/D = 0.5$ until reaching the centerline when the flow is fully developed. This confirms how the nozzle lipline is the zone in which the coflow

starts mixing with the main flow exiting from the nozzle, promoting the arise of turbulence and heat exchange. Finally looking at the u_x and T profiles at $x/D = 5$ and $x/D = 10$, it is evident how they are in the potential core region, indeed the developing boundary layer has still not reached the axis of the flow and there is a flat region where the velocity has the same value of the nozzle exit velocity.



(a) u_x profiles

(b) u_y profiles

(c) k profiles

(d) T profiles

Figure 5.15: Axisymmetric no wall case profiles Mesh 1(Solid line -), Mesh 2 (Dashed line - -), Mesh 3 (Dotted line ··)

5.3 Axisymmetric wall case

Here are reported the results for the simulations considering the wall effect of the duct surrounding the engine. The followed approach is identical to the one described for the no-wall case and the only change is in the definition of geometry and boundary conditions. To account the wall effect provided by the duct, a geometry shape similar to the one provided by the NASA near sonic case has been created. The downstream dimension after the nozzle exit remains the same of the no wall case, while over the nozzle a second geometry part divided in three block has been added. Figure 5.16 shows the geometry dimensions as a function of the nozzle radius for this new case.

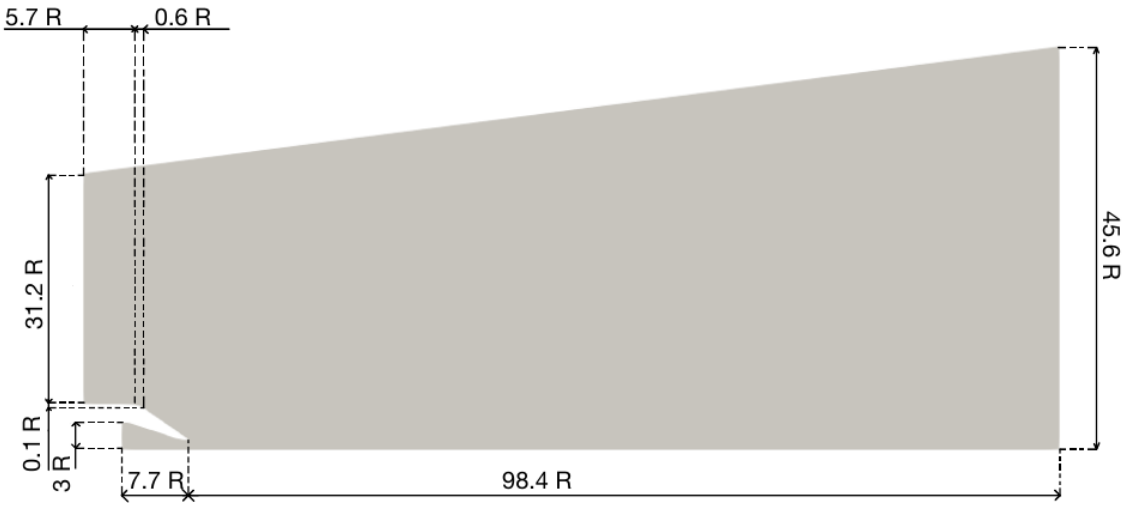


Figure 5.16: Geometry dimensions for the axisymmetric wall case

5.3.1 Mesh generation

As for the no-wall case the mesh generation has been performed using the *BlockMesh* utility. Three meshes with increasing refinement have been created and all of them are structured hexahedral meshes. As for the for the no-wall case the cell expansion in both the axial and radial direction starts from the nozzle lipline following a geometrical progression. The criterion to define the cell size is the same described in Section 5.2.1, the only difference is in the definition of the expansion ratio along x for the duct that has a value smaller than one. This is due because the expansion happens in the opposite direction compared to the x-axis. To have a good refinement in the duct zone, this mesh part has been divided on three blocks following the same strategy adopted for the NASA near sonic validation case (see Figure 4.4). As for the no-wall case, it has been tried to put the majority of the cells along the radial direction in the first two diameters up to the nozzle centerline using an high expansion ratio; this because the jet expansion is strongly limited by the coflow and by the wall at the nozzle exit. Tables 5.6 - 5.8 summarize the mesh parameters for each block while Figure 5.17 represents the most refined mesh front section. The *BlockMesh* file for the most refined grid can be found in Appendix C.

	Mesh 1 127,841 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	110	100	1	1	1
Block 2	57	103	1	0.5	160
Block 3	9	103	1	1	160
Block 4	81	103	1	1	160
Block 5	502	103	1	6	160
Block 6	502	100	1	6	1

Table 5.6: Mesh 1 Axisymmetric wall case

	Mesh 2 203,334 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	110	100	1	1	1
Block 2	57	161	1	0.5	90
Block 3	9	161	1	1	90
Block 4	81	161	1	1	90
Block 5	647	161	1	4	90
Block 6	647	100	1	4	1

Table 5.7: Mesh 2 Axisymmetric wall case

	Mesh 3 318,950 cells				
	N.cells x	N.cells y	N.cells z	Ex. ratio 1	Ex. ratio 2
Block 1	120	110	1	1	1
Block 2	62	215	1	0.5	70
Block 3	10	215	1	1	70
Block 4	88	215	1	1	70
Block 5	838	215	1	3	70
Block 6	838	110	1	3	1

Table 5.8: Mesh 3 Axisymmetric wall case

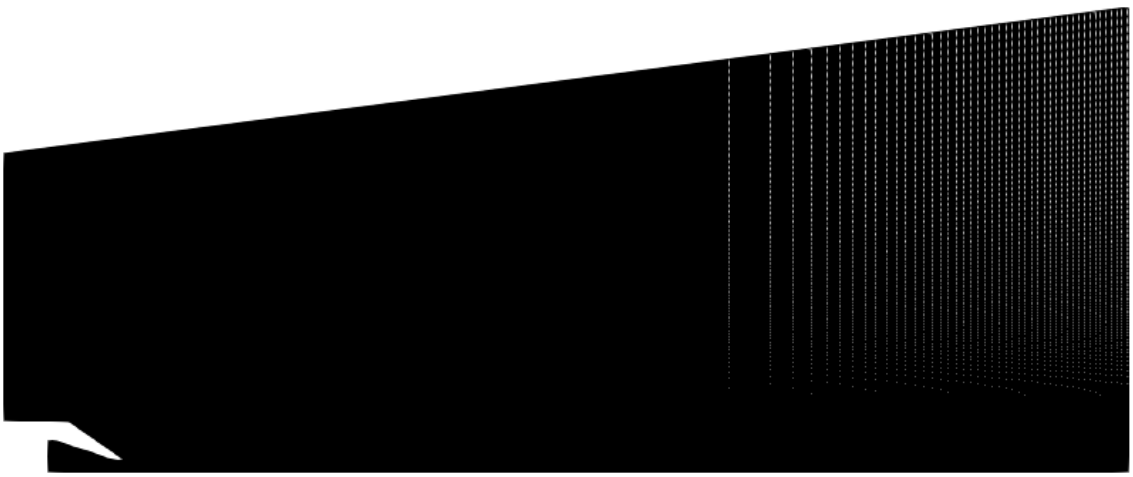


Figure 5.17: Front section Mesh 3 wall case

5.3.2 Boundary and initial condition

The boundary and initial conditions are very similar to the ones specified in the no-wall case. At the nozzle inlet total pressure and total temperature are imposed using (5.3) and (5.4), while for the outlets the *waveTrasmissive* boundary condition remains for both velocity and pressure. The main differences in this case, are the definition of the noSlip boundary condition for both the nozzle and the duct wall and the coflow inlet posed behind the duct wall. Also for this case with all the three generated meshes the $y+$ has a value grater than 10 on both the wall patches, requiring again the the use of the wall functions for the turbulent quantities k, ω, ν_T and α_T. Figure 5.18 shows a simple scheme of the boundary conditions for this new case, while the OpenFOAM files for the boundary and initial conditions can be found in Appendix C.

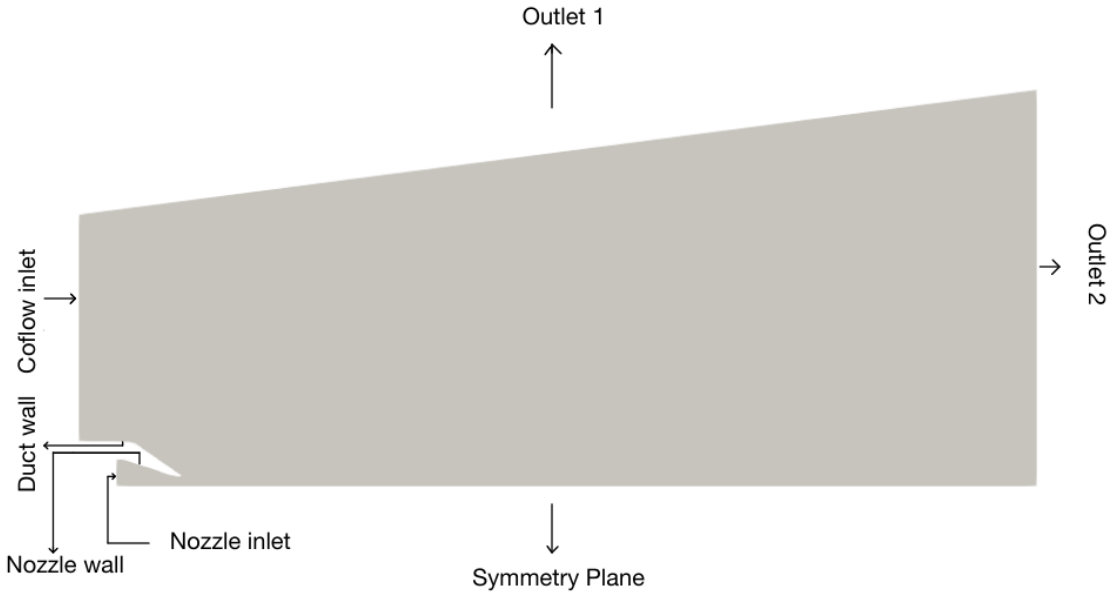
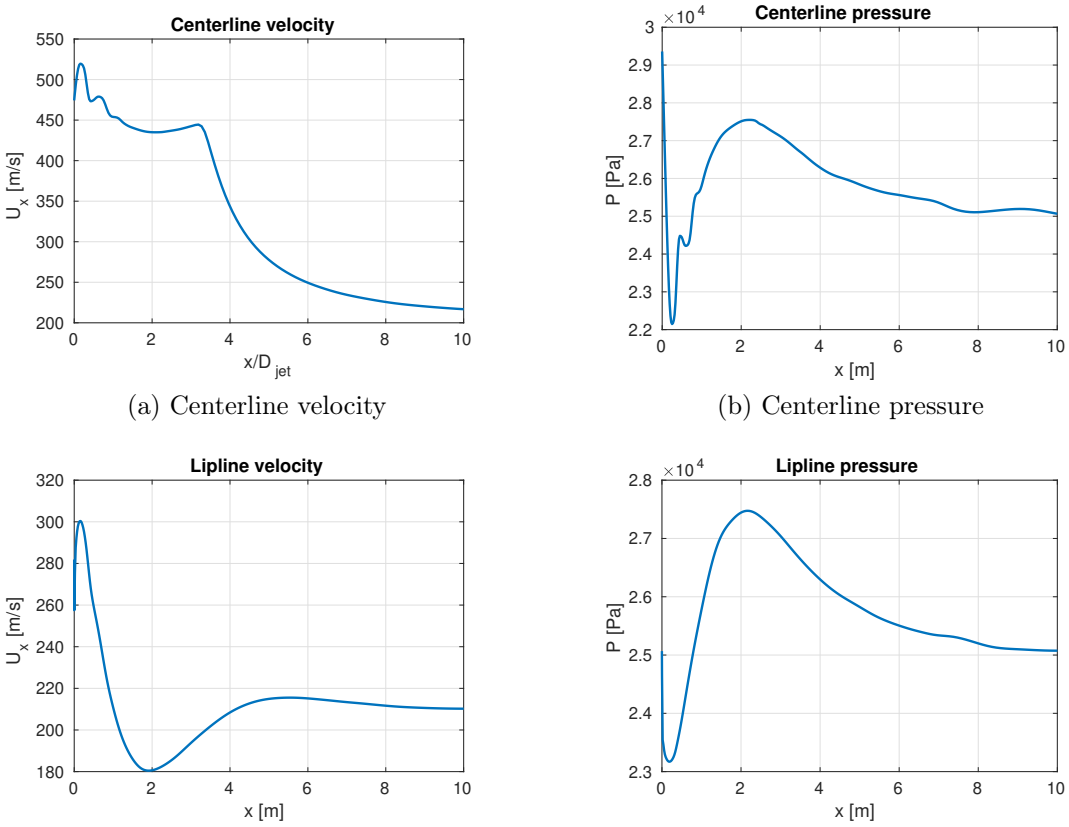


Figure 5.18: Wall case boundary conditions

5.3.3 Results

As for the no-wall the three simulation have been decomposed on 10 cores using the Dragon cluster. The simulation time as well as the maximum Courant number have been set respetively to 1.5 s and 0.5 making the same assumption described in Section 5.2.3. For this case there are no analytical or experimental results through which compare the simulation data. What has been found is that the wall effect of the duct strongly limits the potential core length and it allows the centerline velocity to go below the coflow value, this can be explained looking at the coflow inlet. Starting from the coflow inlet the coflow proceeds along the duct wall and when it reaches the duct corner it starts to expand. The turbulent boundary layer separates and forms a free shear layer in the inclined part of the duct, this creates a low pressure zone that at the beginning allows a further expansion of the main flow exiting from the nozzle. However, when the coflow reattaches to the main flow the pressure have a sudden increase that limits the spreading of the main flow velocity and push back to the inclined duct, part of the flow in the free shear layer, forming

a recirculation zone. The higher pressure strongly limits the centerline velocity out of the nozzle which reaches a value around 210 m/s well below the coflow value of 252 m/s in the free stream over the duct. The formation of the recirculation zone and the decrease in pressure inside it can be observed looking at Figure 5.19 where are reported the evolution along the lipline and centerline of pressure and velocity along the first 10 m of the domain of the Mesh 1.



(a) Centerline velocity

(b) Centerline pressure

(c) Lipline velocity

(d) Lipline pressure

Figure 5.19: Axisymmetric wall case lipline and centerline velocity-pressure evolution

In particular it is interesting to have a look at the quantities along the lipline (Figures 5.19c and 5.19d). Here, it is well evident the pressure drops after few centimeters from the nozzle exit with a consequent velocity peak. After this point, the pressure begins to increase due to the flow reattachment in the recirculation zone above the nozzle lipline, which ends when the pressure reaches its maximum. In this point, the velocity has its lower peak and it can be considered the end of the recirculation zone where the coflow is completely reattached. The same trend of pressure and velocity can be also seen along the centerline (Figures 5.19a and 5.19b), with the only difference that here the velocity is higher, therefore when the pressure drops there are some oscillations due to the formation of shock waves. After the recirculation zone, the pressure restores to the ambient value, however the pressure peak has strongly reduced the velocity of the nozzle flow and of the coflow that start to mix together. This explain why the centerline and lipline velocities have a value below the free stream coflow value of 252 m/s. The distance between the upper and lower peak of the lipline velocity can be used for estimate the length of

the recirculation zone that for this case is of 1.73 m. Figures 5.20 - 5.25 show the the contour plots for the main simulations variables.

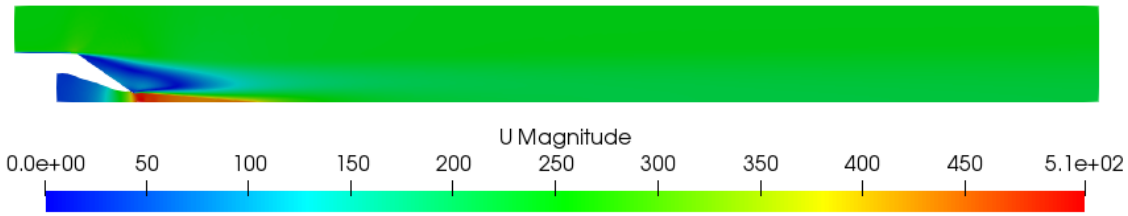


Figure 5.20: U magnitude contour plot wall case

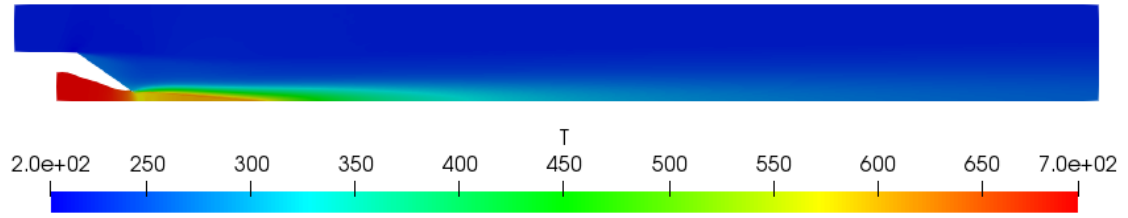


Figure 5.21: T contour plot wall case

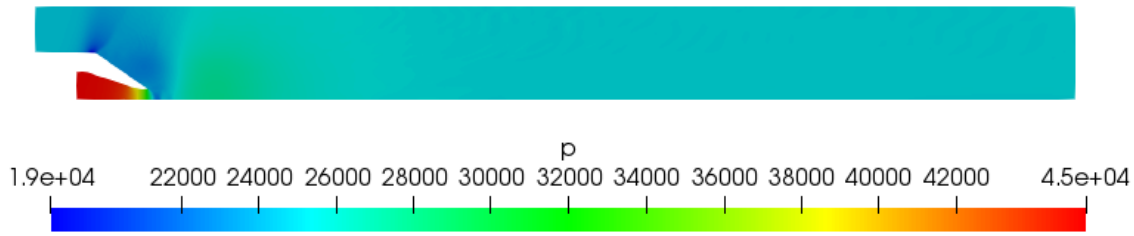


Figure 5.22: p contour plot wall case

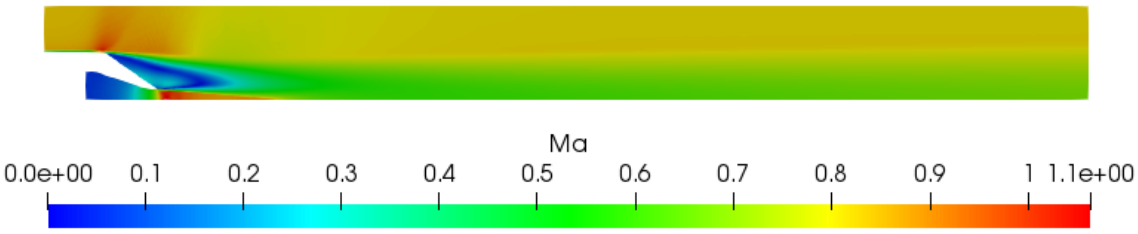


Figure 5.23: Ma contour plot wall case

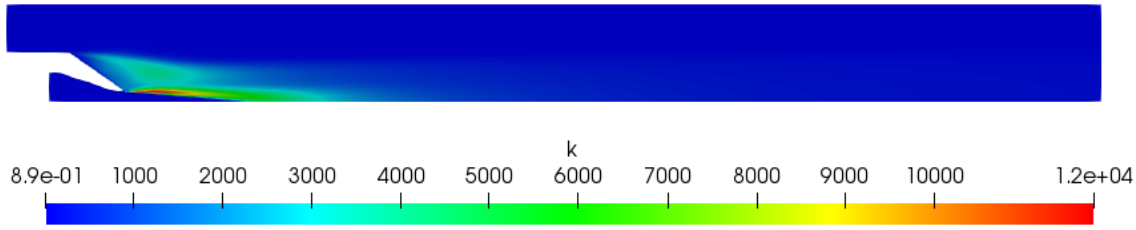


Figure 5.24: k contour plot wall case

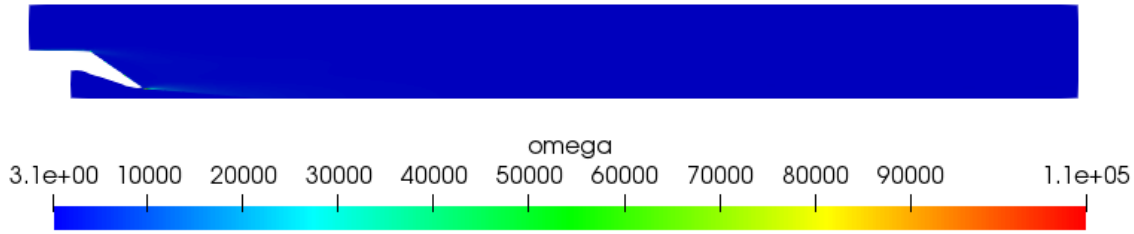


Figure 5.25: ω contour plot wall case

Looking at the contours plot of p (Figure 5.22) and Ma (Figure 5.23), it can be observed how, as the coflow approaches the corner of the inclined part of the duct it starts to expand reaching a Mach value of one, at this point the free shear layer starts to form and the pressure drops. Then over the recirculation zone it is clearly visible the pressure increase with the formation of some shocks close to the nozzle exit. The recirculation can be also identified looking at the contour plot of the turbulent kinetic energy (Figure 5.24), indeed this quantity has an high value not only along the nozzle lipline but also in the inclined part of the duct denoting the formation of turbulence and mixing. The streamlines inside the recirculation zone can be seen in Figure 5.26.

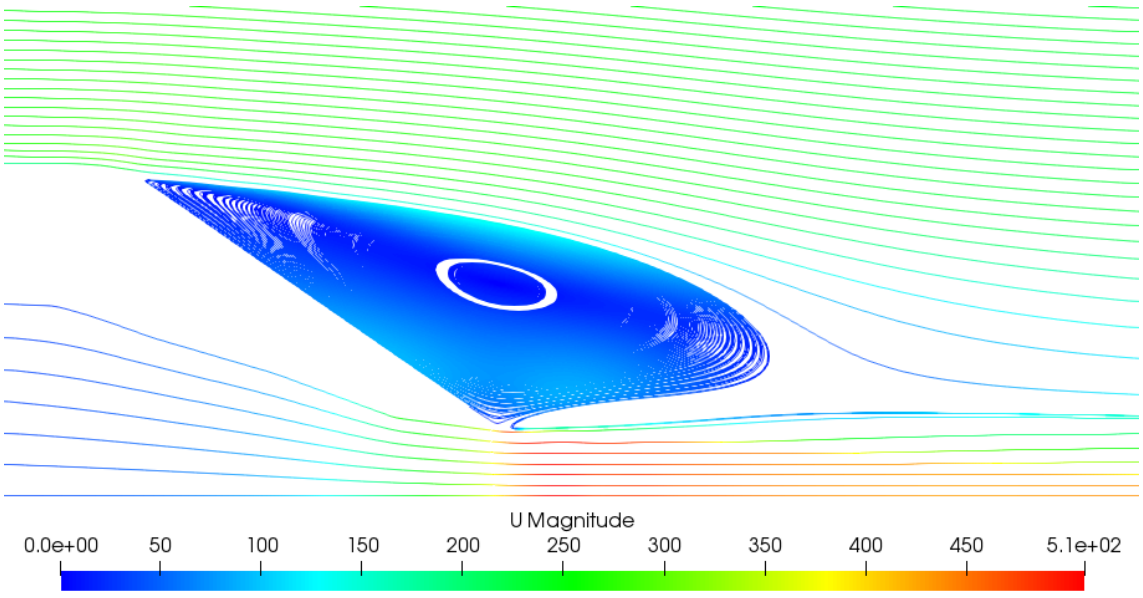
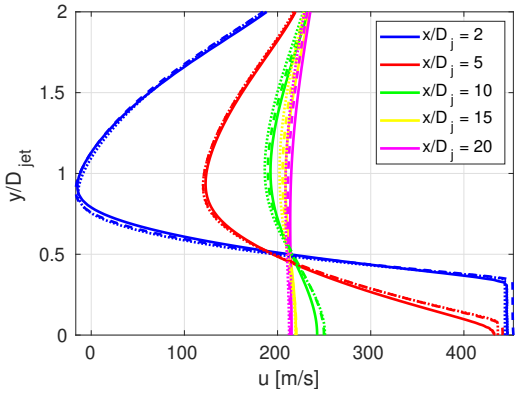
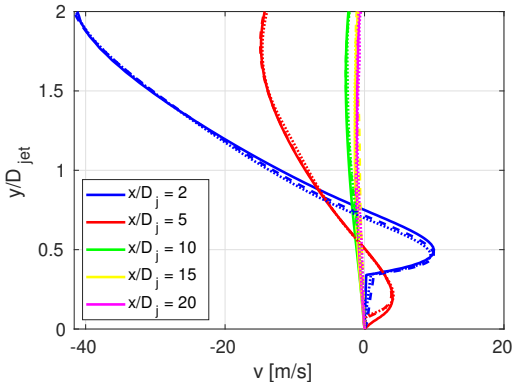


Figure 5.26: Stream lines plot for the recirculation zone

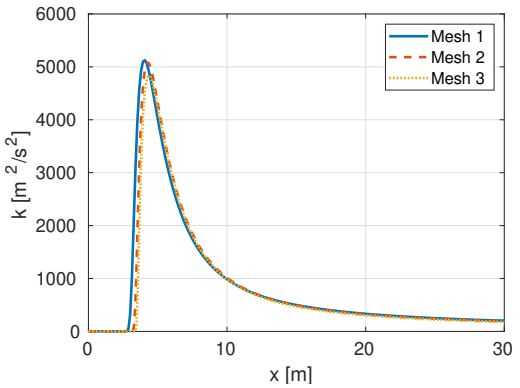
All the results here discussed are for the coarsest grid, as for the no-wall case, to check the grid independence of the results, the velocity, temperature and turbulent kinetic energy profiles along the downstream direction of the nozzle have been plotted. Because the velocity spreading is faster compared to the no-wall case, the profiles are taken along a shorter distance at five locations ($x/D_j = 2, 5, 10, 15, 20$). Looking at figure 5.27 it can been seen that the results do not change appreciably increasing the number of cells in the grids , providing the mesh independence of the results.



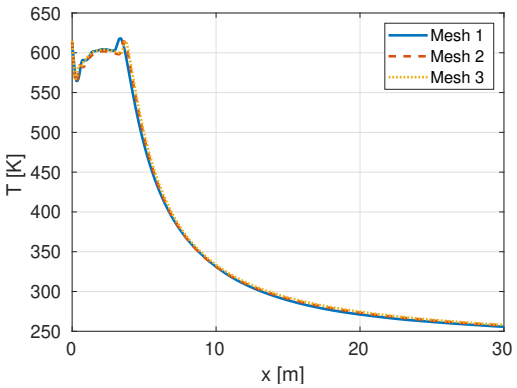
(a) u_x profiles



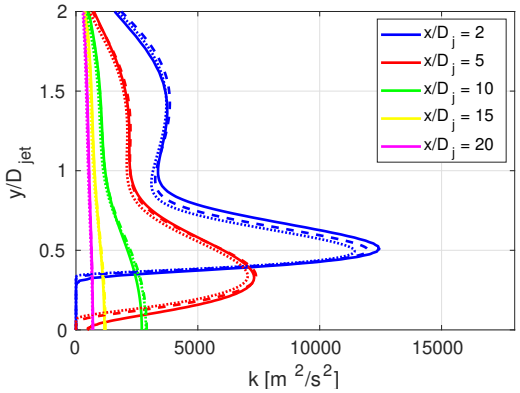
(b) u_y profiles



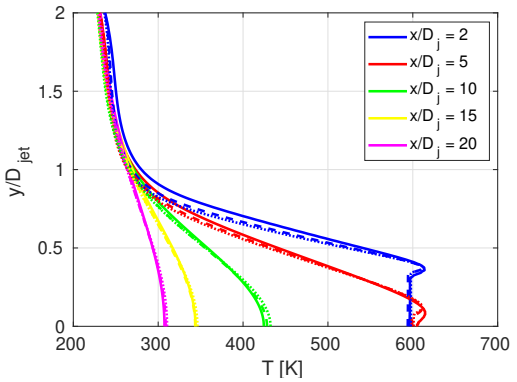
(c) Centerline k



(d) Centerline temperature



(e) k profiles

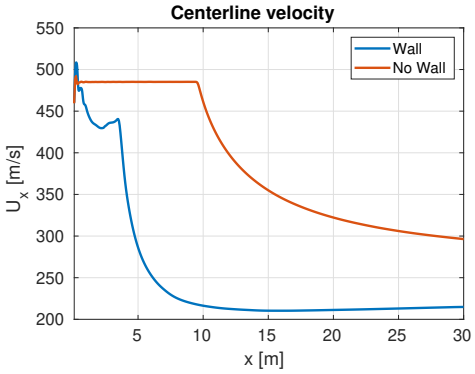


(f) T profiles

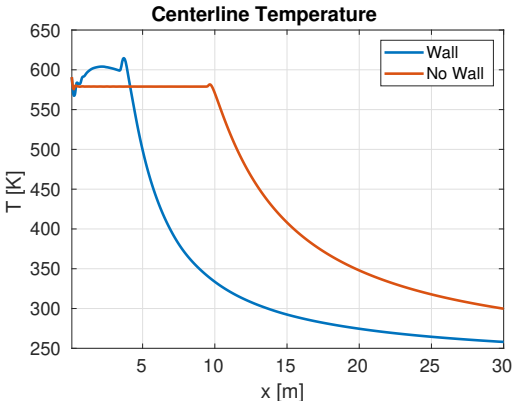
Figure 5.27: Axisymmetric wall case profiles Mesh 1(Solid line -), Mesh 2 (Dashed line - -), Mesh 3 (Dotted line ··)

For the velocity profiles (Figure 5.27a) the same consideration of the no-wall case can be done. In particular it can be noticed how the profiles at $x/D = 2 - 5$ are inside the potential core region where the developing boundary layer has still not reached the axis of symmetry of the jet. However, because of the change of pressure due to the recirculation zone, the potential core has not a constant value of velocity as for the no wall case, but it has higher values of velocity where the pressure is lower and lower values of velocity where the pressure is higher. An opposite trend can be observed instead for the temperature, indeed this quantity directly follows the pressure behavior with higher values at the end of the recirculation zone where the pressure has its peak. Another interesting observation can be done for

the vertical component of the velocity, infact in this case for $x/D = 2 - 5$, which are the downstream location below the recirculation zone, this quantity is bigger of one order of magnitude compared to the no-wall case showing how the mixing is enhanced by the recirculation. Over the potential core zone, the self similarity can be well observed for all the profiles with the turbulent kinetic energy that has lower values compared to the no-wall case due the global lower velocity of the flow. Finally, Figures 6.23a and 6.23b compare the temperature and velocity along the centerline of the two axisymmetric cases here analyzed (results of Mesh 2 for the no-wall case and Mesh 1 for the wall case), while Table 5.9 gives an estimation of the potential core length for all the meshes used to run the simulation, with the percentage reduction of the wall case. In the no-wall case the relative error is computed on the analytical length of the potential core given by (5.7), while in the wall case it is computed on the value of the most refined grid.



(a) Centerline velocity comparison

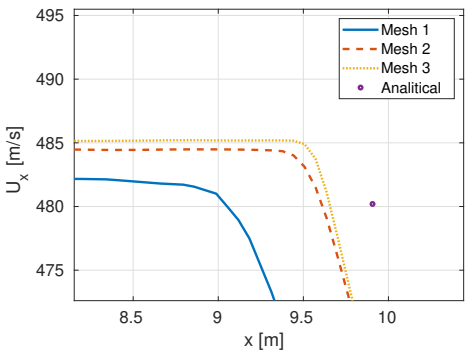


(b) Centerline temperature comparison

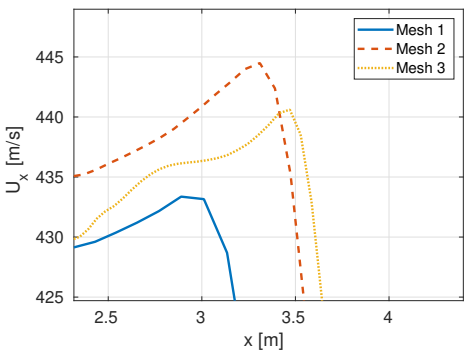
Figure 5.28: Centerline comparisons

	No Wall Case	Error %	Wall Case	Error %	% reduction
Mesh 1	8.988 m	9.25	3.015 m	13.06	66.45
Mesh 2	9.377 m	5.32	3.316 m	4.38	64.64
Mesh 3	9.442 m	4.87	3.468 m	-	63.27

Table 5.9: Potential core comparisons axisymmetric cases



(a) Zoom potential core no-wall case



(b) Zoom potential core wall case

Figure 5.29: Zoom end of the potential core region axisymmetric cases

Chapter 6

Full 3D simulations in flight conditions

In this chapter the flow conditions analyzed in Chapter 5 with the periodic rotating boundary conditions, are now simulated on a full 3D grid. As a first approach this can be seen as a waste of computational resources, indeed the flow is axisymmetric and for a rough analysis a wedge geometry with periodic rotating boundary conditions is more than sufficient to predict the flow's behavior with the RANS model. However, according to the studies of [39], axisymmetric simulations tend to predict a longer length for jet the potential core and can not be used for running Detached Eddy simulations (DES) where the full 3D flow's anisotropies are modeled. Using the same solver set-up described in Chapter 4 and the same methodology discussed in Chapter 5, this last part of the work shows the RANS results of the modeled aircraft jet flow on different 3D grids. Then, the results of the steady state solutions obtained on the most refined grids, are used as initial condition for the DES modeling with the $k-\omega$ *SST DES* turbulence model in order to have an high quality solution, as much as possible similar to the real jet flow physics.

6.1 3D mesh generation using Gmsh

To generate the full 3D mesh it has been decided to not use the build in utility of OpenFOAM *SnappyHexMesh*. This mesh generator needs the .stl cad file of the geometry to mesh and it is mainly used to create unstructured grids of complex geometries. It basically creates the mesh trying to apply subsequent cuts and refinements to a meshed block created using *BlockMesh*, in order to have a meshed body with the same geometry described in the .stl file. Because the geometry involved in this case is axisymmetric and to have accurate results with the DES model it is needed a very fine structured grid, it has been decided to use the Open Source software *Gmsh* for the mesh generation.

6.1.1 Gmsh overview

Gmsh is a three-dimensional finite element mesh generator with a build-in CAD engine ad a post-processor. It is build around four modules: geometry, mesh, solver and post-processing. All geometrical, mesh, solver and post-processing instructions

can be prescribed either interactively using the graphical interface (GUI) or in text files using Gmsh's own scripting language (.geo file). Geometries can be constructed in Gmsh using different CAD kernels, the built-in CAD kernel or the OpenCASCADE kernel. In both of them the definition of the geometry to mesh happens in the same way, first it is needed the definition of the points using the `point` command, then the definitions of curves (using `Line`, `Circle`, `Spline` commands or extruding points), then surfaces (using `Plane Surface` or `Surface` commands, or by extruding curves) and finally volumes (using the `Volume` command or by extruding surfaces). The created geometry entities are named `elementary entities` and each of them need to have an unique tag. The *elementary entities* can be manipulated in various way using the `Translate`, `Rotate`, `Scale` or `Symmetry` command. The Gmsh's mesh module regroups several 1D, 2D and 3D meshing algorithms and the mesh generation happens in a very straight forward way. Curves are discretized first (1D mesh algorithm), the mesh of the curves is then used to mesh the surfaces (2D mesh algorithm), finally the mesh of the surfaces is used to mesh the volumes (3D mesh algorithm). The meshing algorithms can be further divided in *structured* and *unstructured* algorithms. The 2D *unstructured* algorithms generates triangles and/or quadrangles (when the recombination commands or option are used). The 3D *unstructured* algorithms generate tetrahedra, or tetrahedra and pyramids. For all the 2D unstructured algorithms a Delanuy mesh that contains all the points of the 1D mesh is initially constructed using a divide-and-conquer algorithm. After that to generate the final 2D mesh the available algorithms are :

- The **Mesh Adapt** algorithm
- The **Delanuy** algorithm
- The **Frontal Delanuy** algorithm

In general the Mesh Adapt algorithm is the most robust, the Delanuy is the fastest and the Frontal Delanuy is the best in creating high quality cells. In a similar way for the 3D mesh generation the available algorithm are :

- The **Delanuy** algorithm
- The **Frontal** algorithm
- The **HXT** algorithm
- The **MMG3D** algorithm (new and experimental)

Among them the most common used and the most robust is the Delanuy algorithm. The 2D *structured* algorithms (transfinite and extrusion) generate triangles by default but with the command `Recombine` quadrangles can be obtained. The 3D structured algorithms generate tetrahedra, hexahedra, prisms and pyramids, depending on the type of the surface they are based on. The creation of a 3D structured mesh follows the same general principle for the creation of a mesh discussed above, first the curves need to be meshed using the command `Transfinite Curve`, this is followed by the list of the curves on which apply the transfinite algorithm, an expression denoting the number of nodes for each curve and a `Using progression` expression with the common ratio used by the transfinite algorithm to distribute the nodes following a geometrical progression. (Example `Transfinite Curve <curves tags> =`

`<number of nodes> Using progression < common ratio q>`). Defined the transfinite curves it is necessary to define the transfinite surfaces, this happens in a very simple way using the command `Transfinite Surface` followed by the list of the surfaces of which apply the transfinite algorithm. The command can be followed by an optional argument denoting the way the triangles are oriented when the mesh is not recombined. (Example `Transfinite Surface <surface tags> <Left| Right| Alternate| AlternateRight| AlternateLeft>`). Finally it is the time to define the transfinite algorithm on the geometry's volumes. This can happen in different ways, using the command `Transfinite Volume`, followed by the tag of the volume on which apply the transfinite algorithm and the list of the volume geometry points ordered in an counterclockwise direction (Example `Transfinite Volume <volume tag> = <list of volume's points>`); or using the `Extrude` command inside the Gmsh geometry module. With this command it is possible to generate a geometry volume on which is already implemented the transfinite algorithm, starting from the transfinite surfaces. The extrusion can be a rotation or a translation, and in both cases it is necessary to define the direction along which rotate or translate the transfinite surfaces, the surfaces to extrude and the number of layers to create during the extrusion. The layers can be put in a geometrical progression, using the `Using Progression` option followed by the expression for the common ratio q.
About the solver and the post-processing module, the former it is used to drive external solver and codes on Gmsh through the ONELAB interface, the latter it is used to post-process simulation data obtained using external software on the mesh created in the Gmsh environment. Further information about the Gmsh software and its use can be found at [40] and on the official Gmsh website [41].

6.1.2 Mesh creation

The mesh generation has been done recreating as a first step the 2D geometry in the Gmsh environment for both the wall and the no-wall case, but with a small change in the vertical length of the domain front section. Indeed, because the 3D structured mesh is created using the `Extrude` command with a rotation of the domain front section, it is necessary to reduce the height of the domain, in order to have a small cylinder hole along the domain centerline after the extrusion procedure. This is necessary because, if the 2D geometry is left unchanged and the extrusion has as its center the bottom left corner of the 2D geometry, a singularity with no cells is created along the domain centerline. This does not allow the solver to compute the gradients along this direction and hence to properly discretize the model equations. The cylinder hole is then filled with a squared base parallelepiped, with the square side $\sqrt{2}$ smaller than the hole radius. The radius of the cylinder's base can be varied in order to have a finer or coarser meshes along the centerline. In this way the discretization of the equations along the axis of the 3D geometry is guaranteed. This mesh procedure is well known for all the full 3D axisymmetric problems and takes the name of *butterfly* grid generation. In Gmsh the extrusion of the base 2D geometry has been done for 4 times along an angle of 90°.The base geometry has exactly the same block definitions and expansion ratios for the axisymmetric cases analyzed in Chapter 5, the only difference is that in Gmsh rather than define the expansion ratios, it necessary to define the common ratio q which is related

to the expansion ratio through (4.21). Five meshes divided on two groups have been created for both the wall and the no wall cases. In the first mesh group the radial and axial mesh refinements have been increased leaving the extrusion angle constant to a value of 10°. This group is composed of three meshes with the half-section refinement of Mesh 1, Mesh 2 and Mesh 3 of the axisymmetric wall and no wall cases. Contrarily, in the second group, the axial mesh refinement has been left unchanged while the extrusion angle has been gradually reduced to 10°, 5° and 2°. For the no-wall case it has been decided to use the half-section refinement of Mesh 2 of the axisymmetric case, while for the wall case it has been decided to use the half section refinement of Mesh 1 of the axisymmetric case. Obviously, from what it has been said it is clear that one mesh is in common between the two groups. Tables 6.1 - 6.2 summarize the mesh data for the full 3D geometry of both the no-wall and wall cases.

	Group 1 - axial and radial refinement		
	Cells in the half-section	Extrusion angle	Total number of cells
Mesh 2.5	63,194	10°	2,234,295
Mesh 5M	139,927	10°	4,970,945
Mesh 8M	202,462	10°	7,212,501

(a) Group 1 summary table

	Group 2 - angular refinement		
	Cells in the half-section	Extrusion angle	Total number of cells
Mesh 10°	139,927	10°	4,970,945
Mesh 5°	139,927	5°	9,897,228
Mesh 2°	139,927	2°	23,662,125

(b) Group 2 summary table

Table 6.1: Meshes for the full 3D no-wall case

	Group 1 - axial and radial refinement		
	Cells in the half-section	Extrusion angle	Total number of cells
Mesh 4M	127,841	10°	4,026,134
Mesh 8M	202,334	10°	8,088,912
Mesh 12M	319,950	10°	12,265,040

(a) Group 1 summary table

	Group 2 - angular refinement		
	Cells in the half-section	Extrusion angle	Total number of cells
Mesh 10°	127,841	10°	4,026,134
Mesh 5°	127,841	5°	8,052,268
Mesh 2°	127,841	2°	20,130,670

(b) Group 2 summary table

Table 6.2: Meshes for the full 3D wall case

The great amount of generated meshes has as main objective to show the grid independence of the solution varying the mesh refinement in the three coordinates of the domain. As a general target in the mesh generation, it has been tried to stay around the range of 20 millions cells for the most refined mesh in order to have a reasonable computational cost. This explains why the number of cells in Mesh 2 and Mesh 1 of the axisymmetric no-wall and wall cases have been chosen as half-section for the angular mesh refinement. Indeed, extruding these half sections every 2° leds to a 3D mesh of 23,662,125 cells for the no-wall case and 20,130,670 cells for the wall case. Moreover, the solutions obtained with the periodic-rotating boundary conditions on these two-half section is not far from the ones obtained with the most refined grid (Mesh 3). This suggest that this mesh configuration has the best trade off between computational cost and numerical accuracy. The generated meshes have been exported in the .msh format and then converted to the OpenFOAM's format using the mesh conversion utility *gmshToFoam*. Figure 6.1 and Figure 6.2 shows the mesh sections for the most refined cases while Figure 6.3 presents the butterfly grid strategy used for the mesh generation. The gmsh .geo file for the most refined mesh generation of both the wall and no-wall cases can be found in AppendixD.

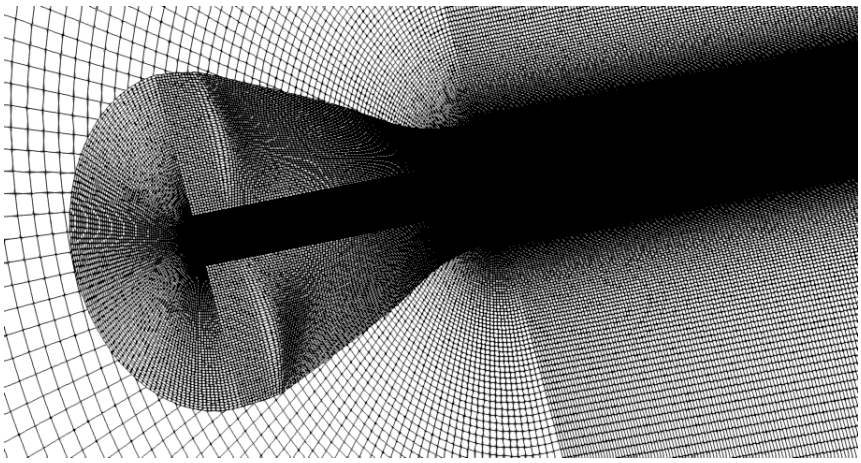


Figure 6.1: 3D mesh no-wall case

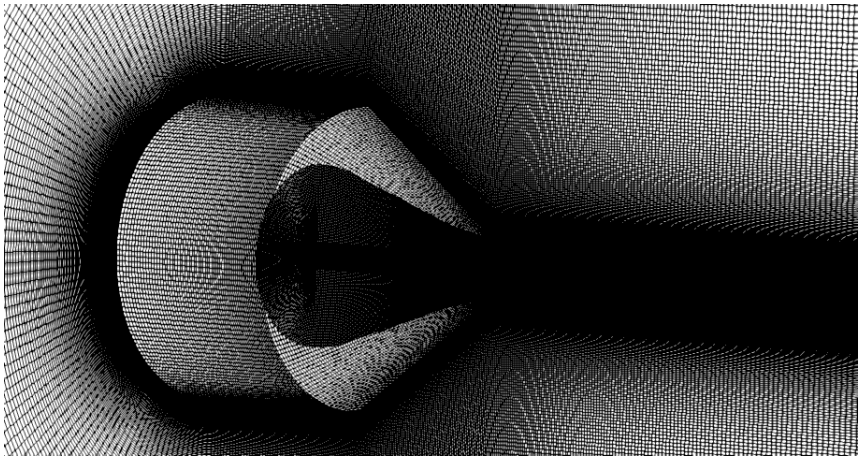


Figure 6.2: 3D mesh wall case

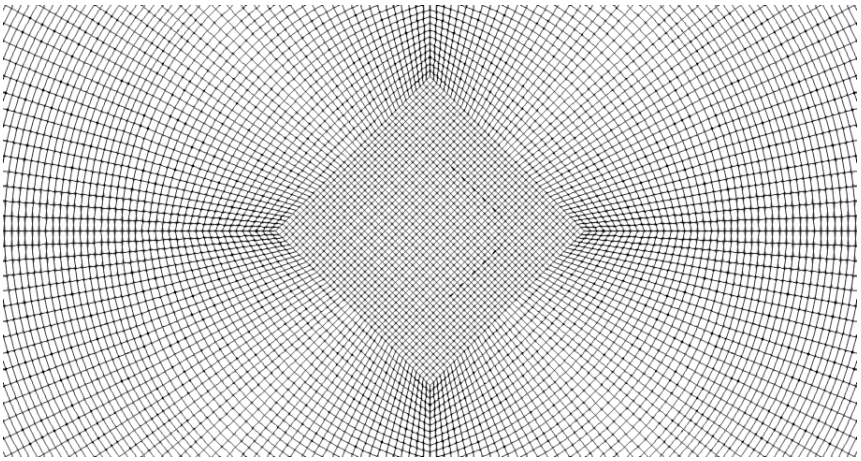
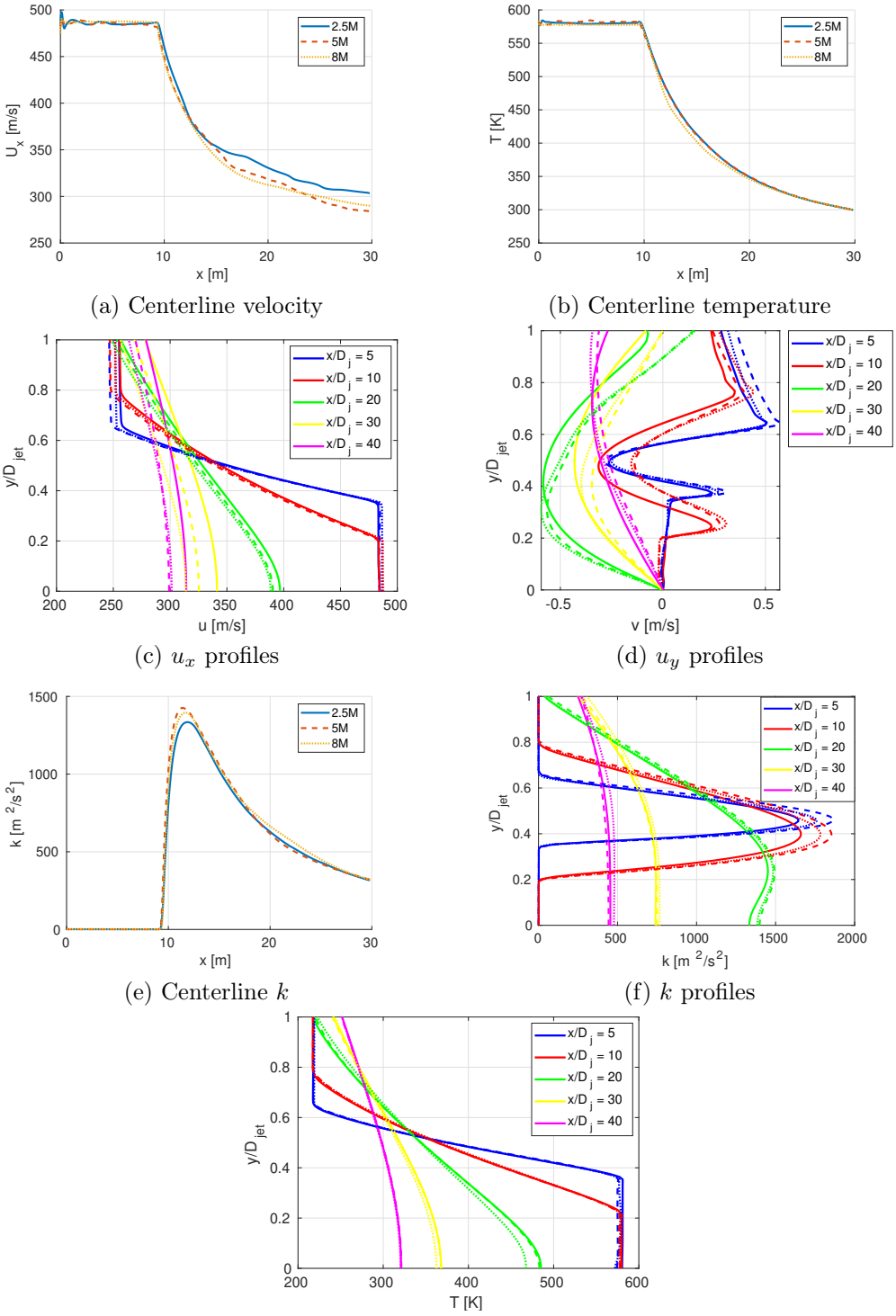


Figure 6.3: Buttefly grid strategy

6.2 Full 3D no wall case - RANS approach

In this section are reported the results for the full 3D no wall cases. For both the no-wall and wall cases the simulation controls, boundary and initial conditions are the same of the corresponding axisymmetric cases.(see Sections 5.2.2, 5.2.3 and 5.3.2). The only clear difference is that because they are full 3D cases the boundary conditions of wedge and symmetryPlane are absent. The most refined cases of 23 million cells (no-wall case) and 20 million cells (wall case) have been run on Theta, the supercomputer of the Argonne National Laboratory on 4096 cores. Theta is a Cray XC40 machine based on second-generation of the Intel Xeon Phi processor, it has 4,392 computers nodes with 64 processors for each node and it can reach a maximum velocity of 11.69 petaflops. More information about Theta and how to get an user account on this machine can be found at [42]. All the other simulations have been run on 40 cores using Dragon. First are reported the results for the first mesh group in which it has been changed the axial and radial mesh refinement, then follows the results for the second group in which it has been varied the extrusion angle. Finally a comparison between the axisymmetric case and the full 3D case is made.

6.2.1 Group1 - results



(a) Centerline velocity

(b) Centerline temperature

(c) u_x profiles

(d) u_y profiles

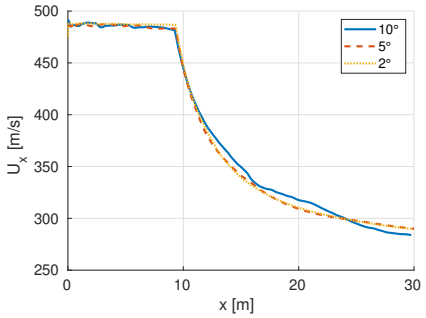
(e) Centerline k

(f) k profiles

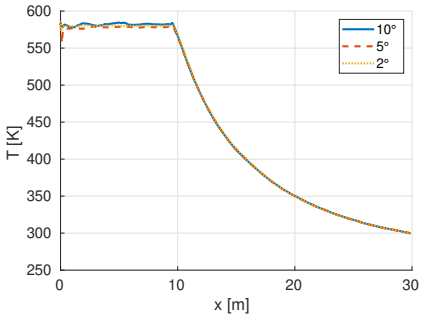
(g) T profiles

Figure 6.4: Results comparisons Group 1 no-wall case Mesh 2.5M (solid line -), Mesh 5M (dashed line - -), Mesh 8M (dotted line :)

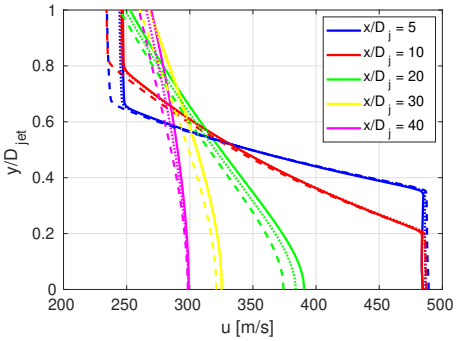
6.2.2 Group2 - results



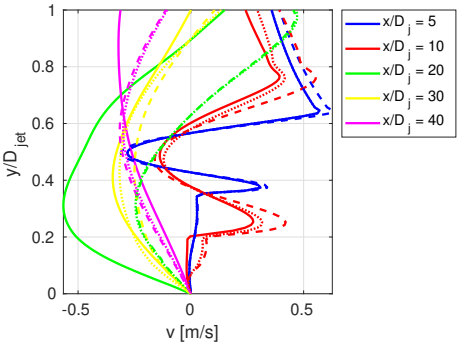
(a) Centerline velocity



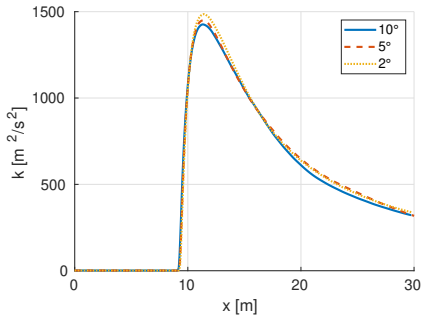
(b) Centerline temperature



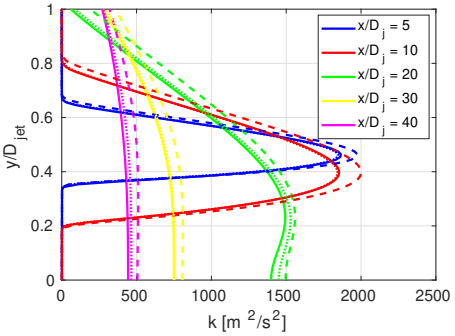
(c) u_x profiles



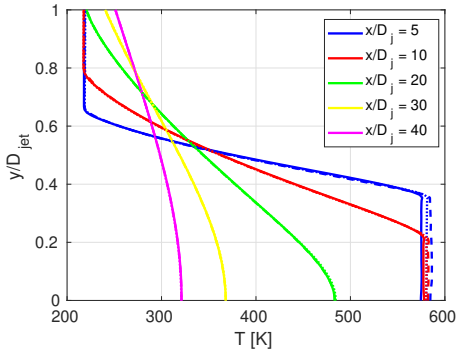
(d) u_y profiles



(e) Centerline k



(f) k profiles



(g) T profiles

Figure 6.5: Results comparisons Group 2 no-wall case Mesh 10° (solid line -), Mesh 5° (dashed line - -), Mesh 2° (dotted line :)

From Figures 6.4 - 6.5 it is evident that the simulation solutions converge quite well for both the groups of meshes. It is interesting to notice how increasing the number of points, the solution for the centerlines velocity and temperature in the jet potential core becomes flatter. It is in this region, where the flow is near $Ma = 1$, that numerical instabilities, due to a coarse grid, may arise. The only variable that seems to not converge well is the vertical component of the velocity, indeed for the less refined cases the profiles for this velocity component, seems to vary a lot especially far from the nozzle exit. However, between the simulation variables, it is the one with the lower order of magnitude and the pour agreement between the the different grids does not affect the validity of the simulation. The same flow characteristic of the axisymmetric case, as the self-similarity of the simulation profiles, the developing of the boundary layer for the velocity and temperature profiles in the potential core region and the turbulence that starts to arise from the jet lipline can be well seen also for this 3D case. To check the grid convergence of the results, Table 6.3 shows the jet potential core length and its relative error one the value obtained with the analytical relation (5.7), while Figures 6.6-6.10 show the contour plots of the main simulation variables obtained on the finest grid of 23,662,125 cells (Mesh 2° of group 2).

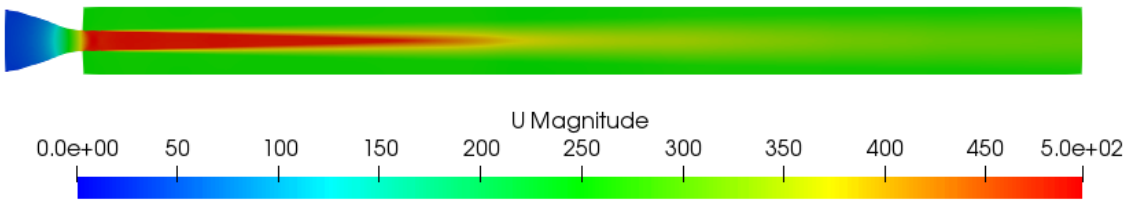


Figure 6.6: U contour plot 3D no-wall case

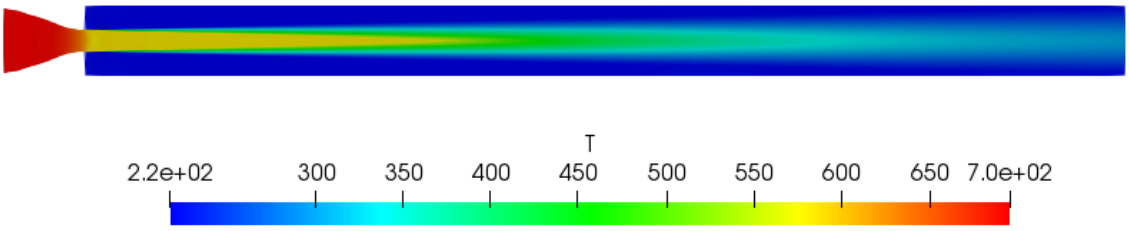


Figure 6.7: T contour plot 3D no-wall case

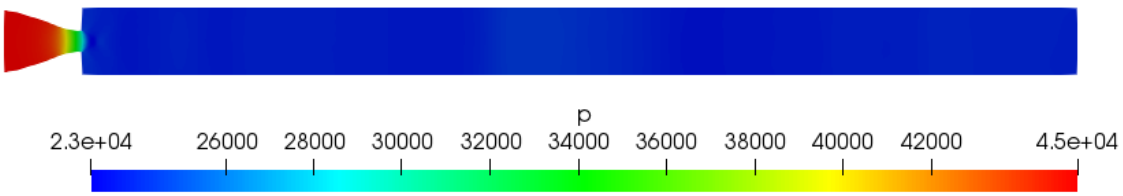


Figure 6.8: p contour plot 3D no-wall case

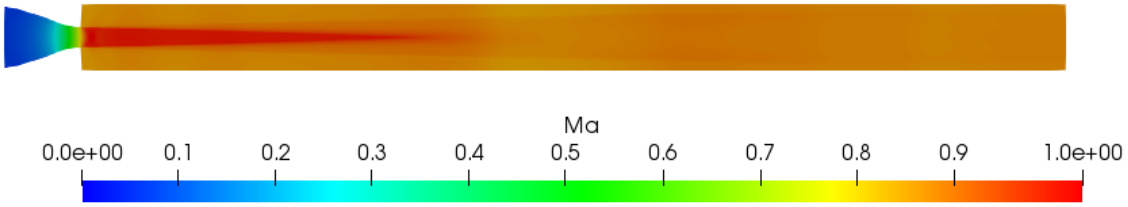


Figure 6.9: Ma contour plot 3D no-wall case

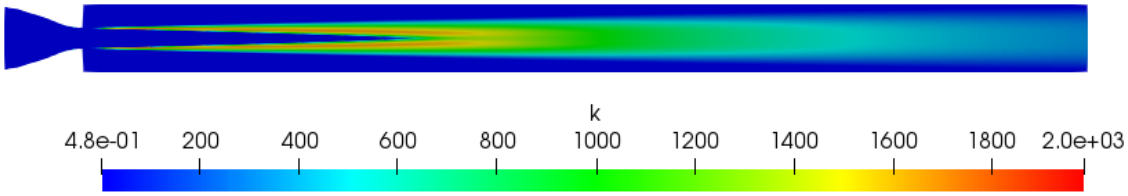


Figure 6.10: k contour plot 3D no-wall case

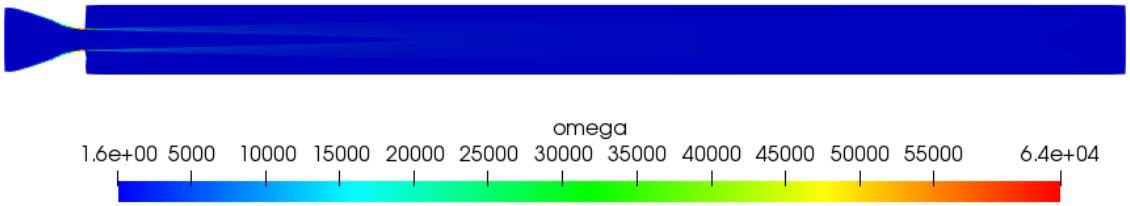


Figure 6.11: ω contour plot 3D no-wall case

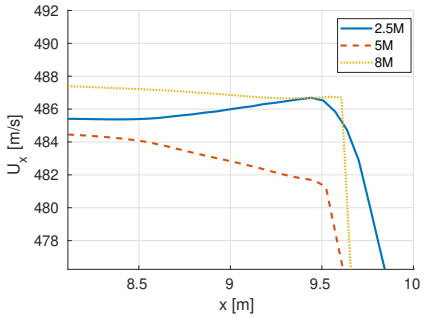
	Group 1	
	P.C. length [m]	Error %[-]
Mesh 2.5M	9.507	4.00
Mesh 5M	9.522	3.85
Mesh 8M	9.607	3.00

(a) Group 1 - axial and radial refinement

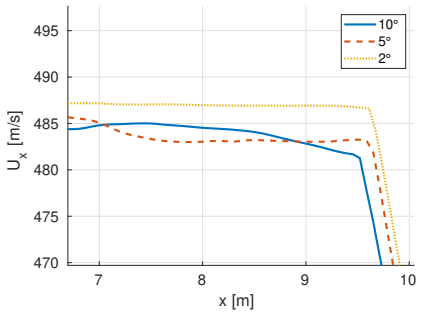
	Group 2	
	P.C. length [m]	Error %[-]
Mesh 10°	9.522	3.85
Mesh 5°	9.587	3.20
Mesh 2°	9.613	2.94

(b) Group 2 - angular refinement

Table 6.3: Grid convergence potential core - no wall case



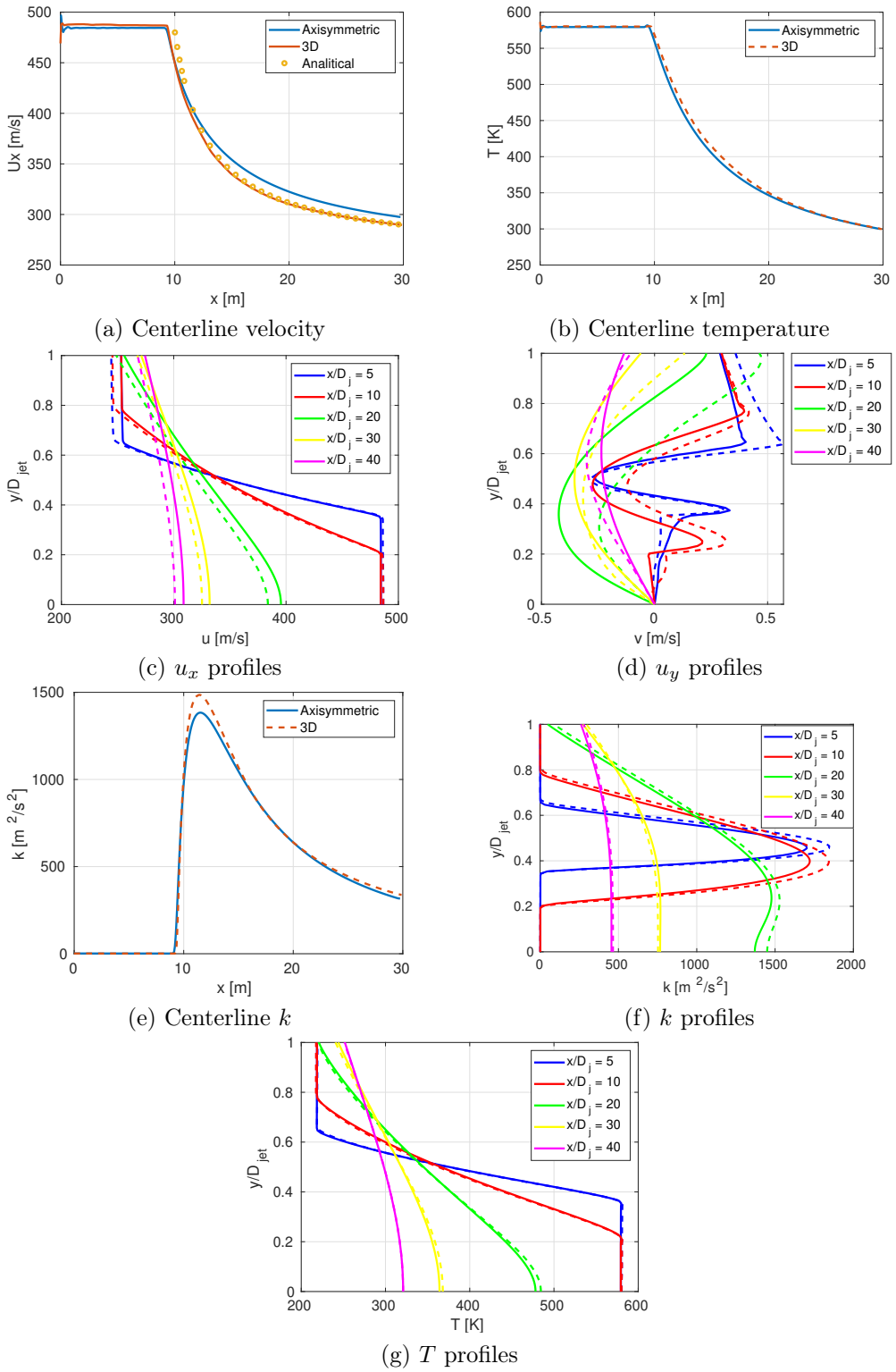
(a) Zoom potential core no-wall case - Group 1



(b) Zoom potential core no-wall case - Group 2

Figure 6.12: Zoom end of the potential core region 3D no-wall cases

6.2.3 Comparisons Axisymmetric-3D



(a) Centerline velocity

(b) Centerline temperature

(c) u_x profiles

(d) u_y profiles

(e) Centerline k

(f) k profiles

(g) T profiles

Figure 6.13: Comparisons between the 3D axisymmetric and the full 3D no-wall cases. 3D axisymmetric (Solid line -) full 3D (Dashed line - -)

Figure 6.13 shows the comparisons between the results obtained with Mesh 2 (axisymmetric) and Mesh 2° (full 3D) of the no-wall cases . As can be seen there are no big differences between the results of the two simulations, but the full 3D case shows a better agreement with the analytical solution provided by (5.6). In particular, the full 3D case shows an higher turbulent kinetic energy with a consequent higher turbulent viscosity ν_T that makes the flow more diffusive compared to the axisymmetric case. In the far field, far from the nozzle exit, both the the axisymmetric and 3D curves becomes parallel demonstrating the same centerline spreading for both the simulations. Regarding the jet potential core length, it is slightly shorter in the axisymmetric case compared to the full 3D case (9.377 m axisymmetric - 9.613 m 3D), this contradicts what has been stated in [39]. A further proof can be given considering that the 3D case predicts an higher centerline velocity exiting from the nozzle compared to the axisymmetric case (3D 486.6 m/s, axisymmetric 484.4 m/s) and from (5.7), it is evident that the higher is the difference between U_o and U_a, the shorter is the potential core length.

6.3 Full 3D wall case - RANS approach

Following the same order of the no-wall case here are reported the results for the full 3D simulations of the wall case. Also in this case, varying the mesh refinement, the solution continues to converge well and the observations done for the axisymmetric case continue to be valid for the 3D case (See Figures 6.15 and 6.16). Figures 6.17 - 6.22 show the contour plots of the main simulation variables for the most refined case, while it is interesting to look at Figure 6.14 where are represented the recirculation streamlines along the full 3D geometry. In particular in the full 3D case, it is well evident the formation of the free shear layer in the inclined part of the nozzle duct which allows the further expansion of the flow exiting from the nozzle.

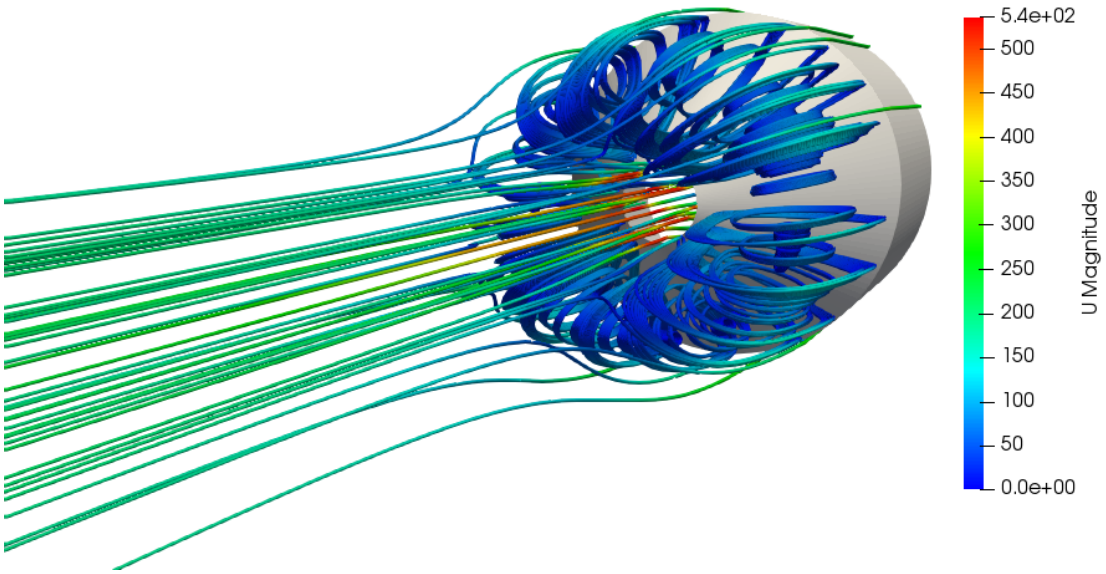
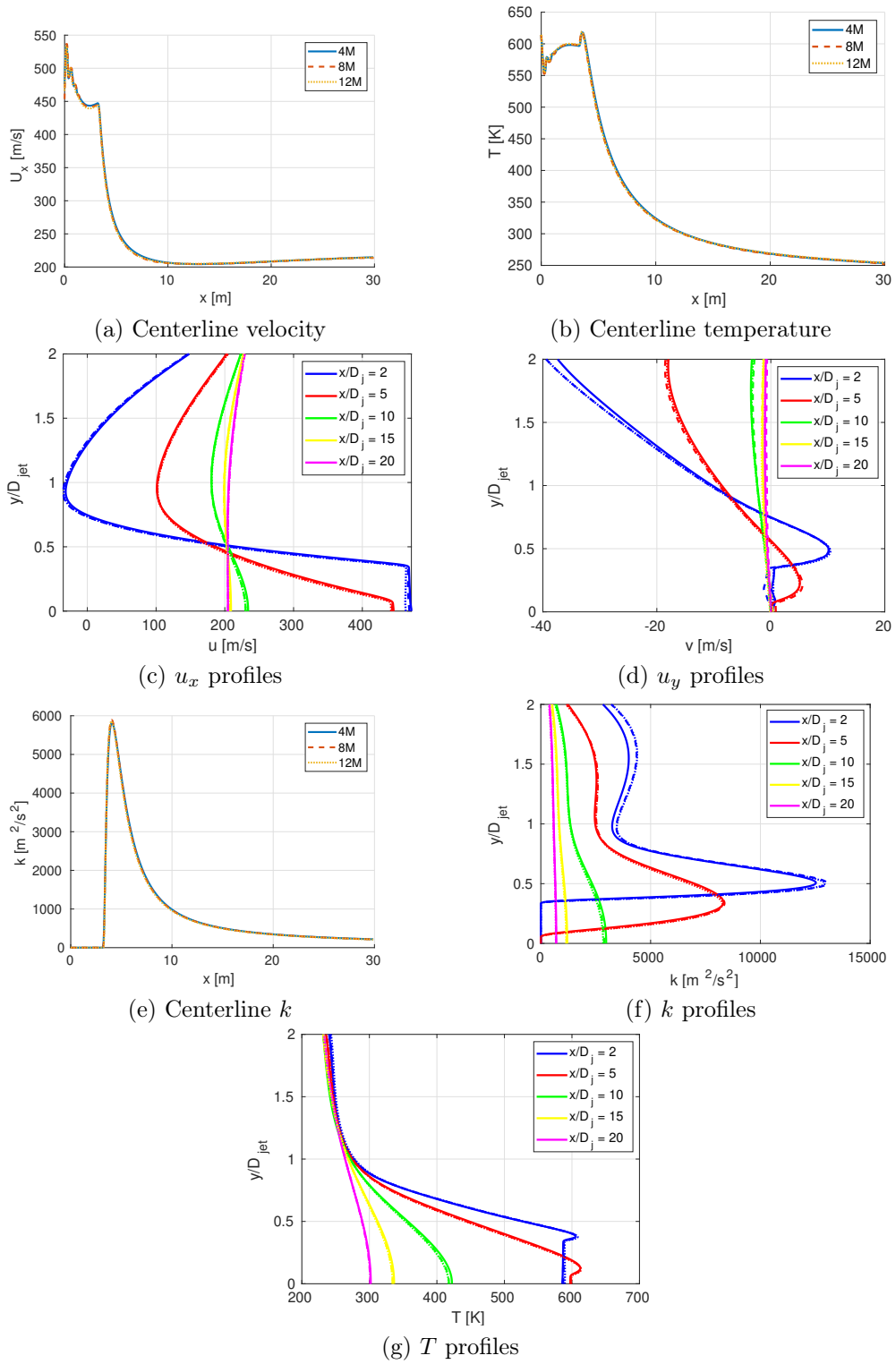


Figure 6.14: 3D streamlines wall case - RANS approach

6.3.1 Group 1 - results



(a) Centerline velocity

(b) Centerline temperature

(c) u_x profiles

(d) u_y profiles

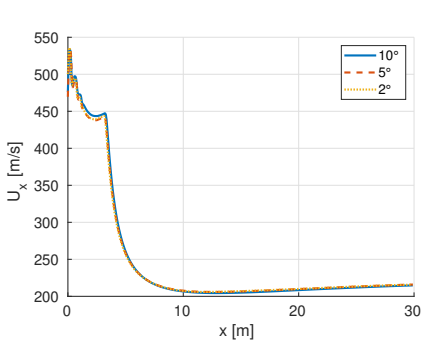
(e) Centerline k

(f) k profiles

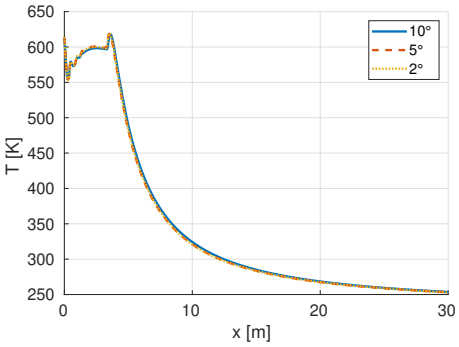
(g) T profiles

Figure 6.15: Results comparisons Group 1 wall case Mesh 4M (solid line -), Mesh 8M (dashed line - -), Mesh 12M (dotted line :)

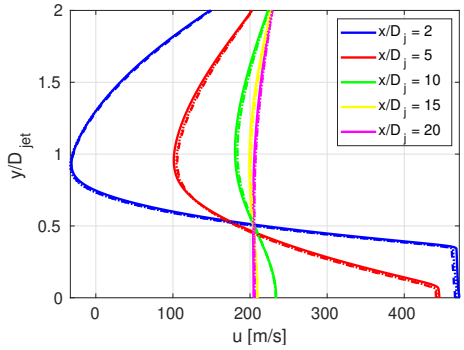
6.3.2 Group 2 - results



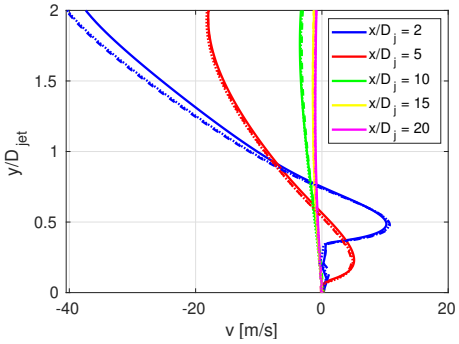
(a) Centerline velocity



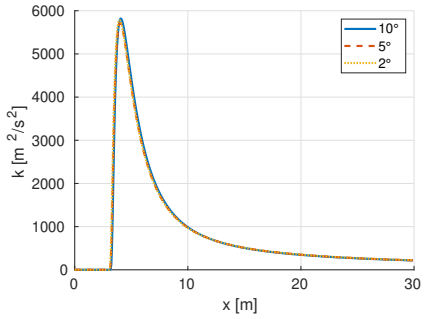
(b) Centerline temperature



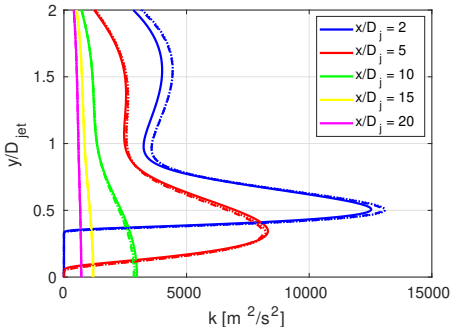
(c) u_x profiles



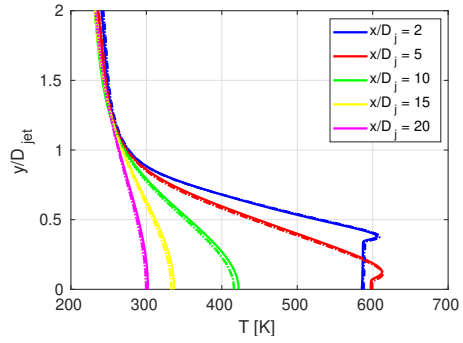
(d) u_y profiles



(e) Centerline k



(f) k profiles



(g) T profiles

Figure 6.16: Results comparisons Group 2 wall case Mesh 2° (solid line -), Mesh 5° (dashed line - -), Mesh 10° (dotted line :)

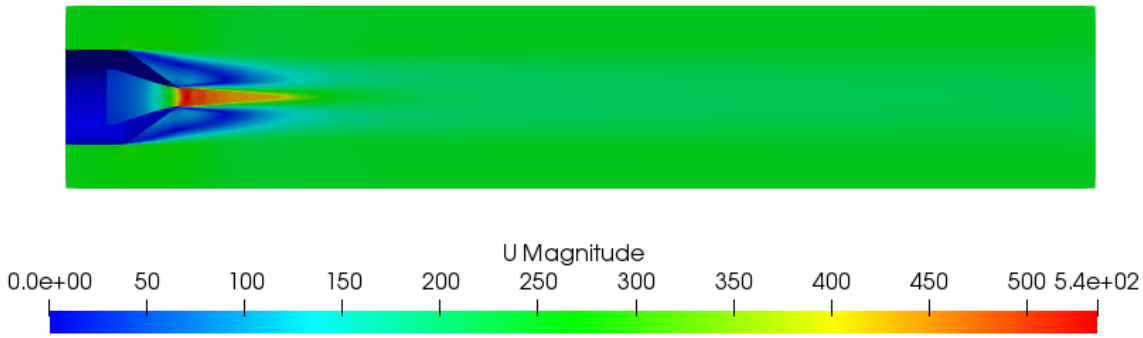


Figure 6.17: U contour plot 3D wall case

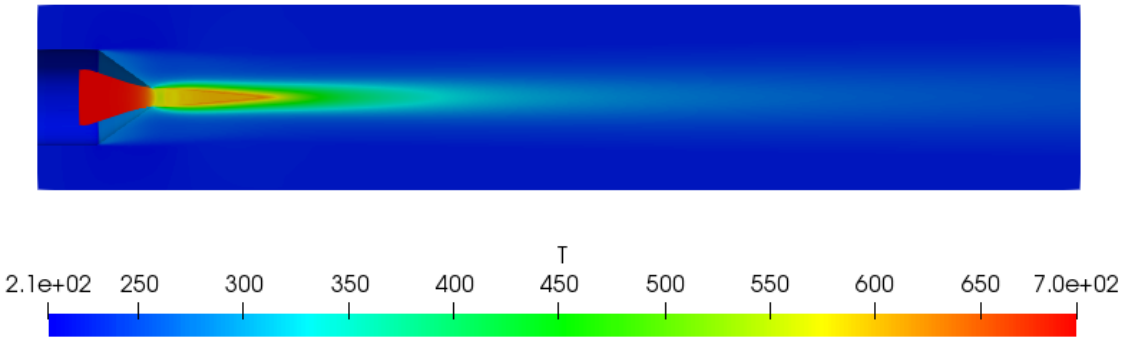


Figure 6.18: T contour plot 3D wall case

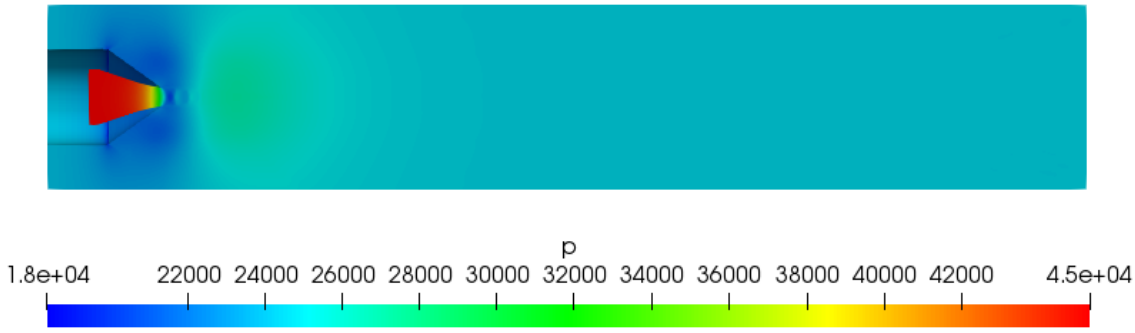


Figure 6.19: p contour plot 3D wall case

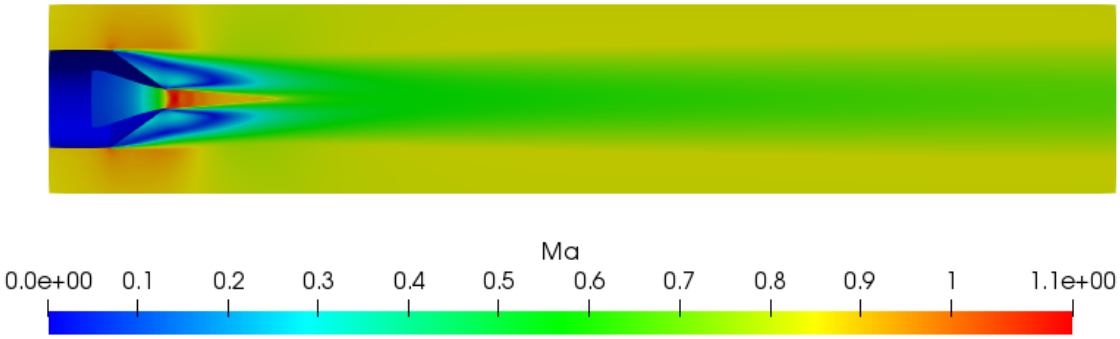


Figure 6.20: Ma contour plot 3D wall case

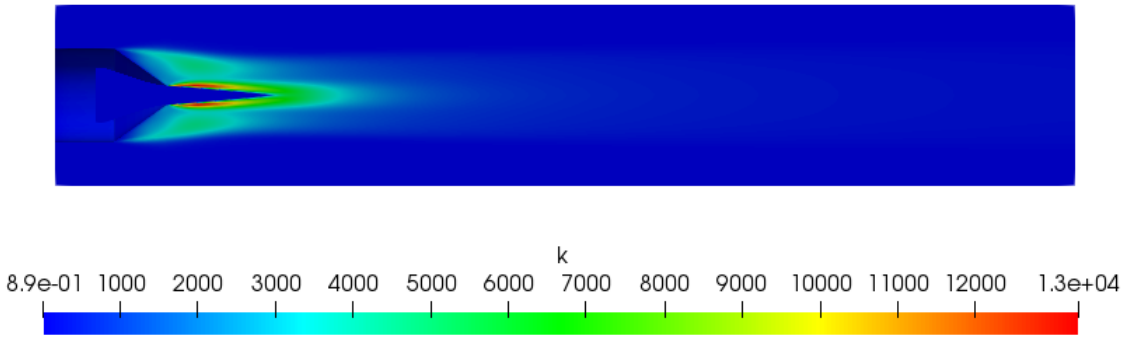


Figure 6.21: k contour plot 3D wall case

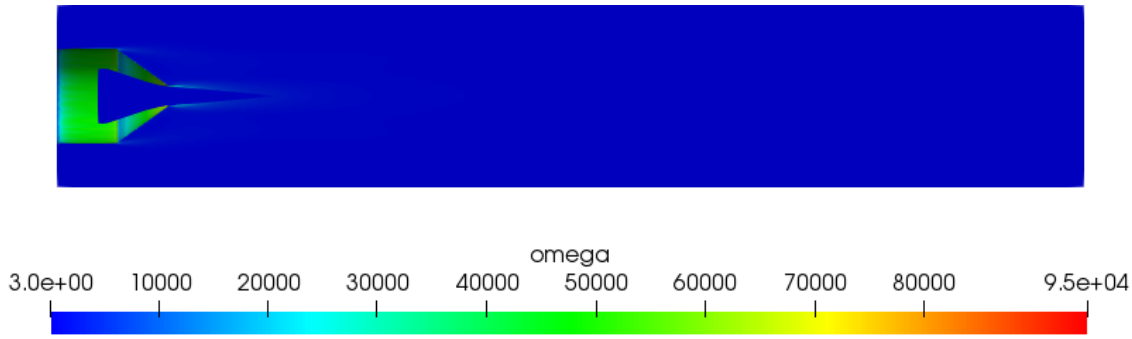


Figure 6.22: ω contour plot 3D wall case

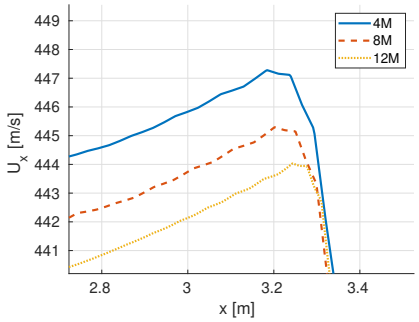
	Group 1	
	P.C. length [m]	Error %[-]
Mesh 4M	3.236	1.22
Mesh 8M	3.250	0.79
Mesh 12M	3.276	-

(a) Group 1 - axial and radial refinement

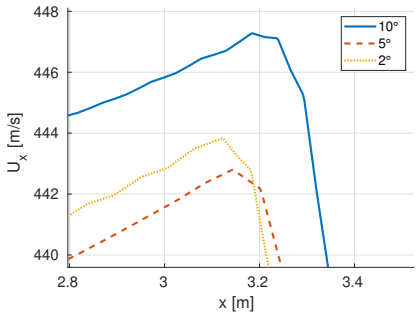
	Group 2	
	P.C. length [m]	Error %[-]
Mesh 10°	3.236	1.85
Mesh 5°	3.198	0.66
Mesh 2°	3.177	-

(b) Group 2 - angular refinement

Table 6.4: Grid convergence potential core - wall case



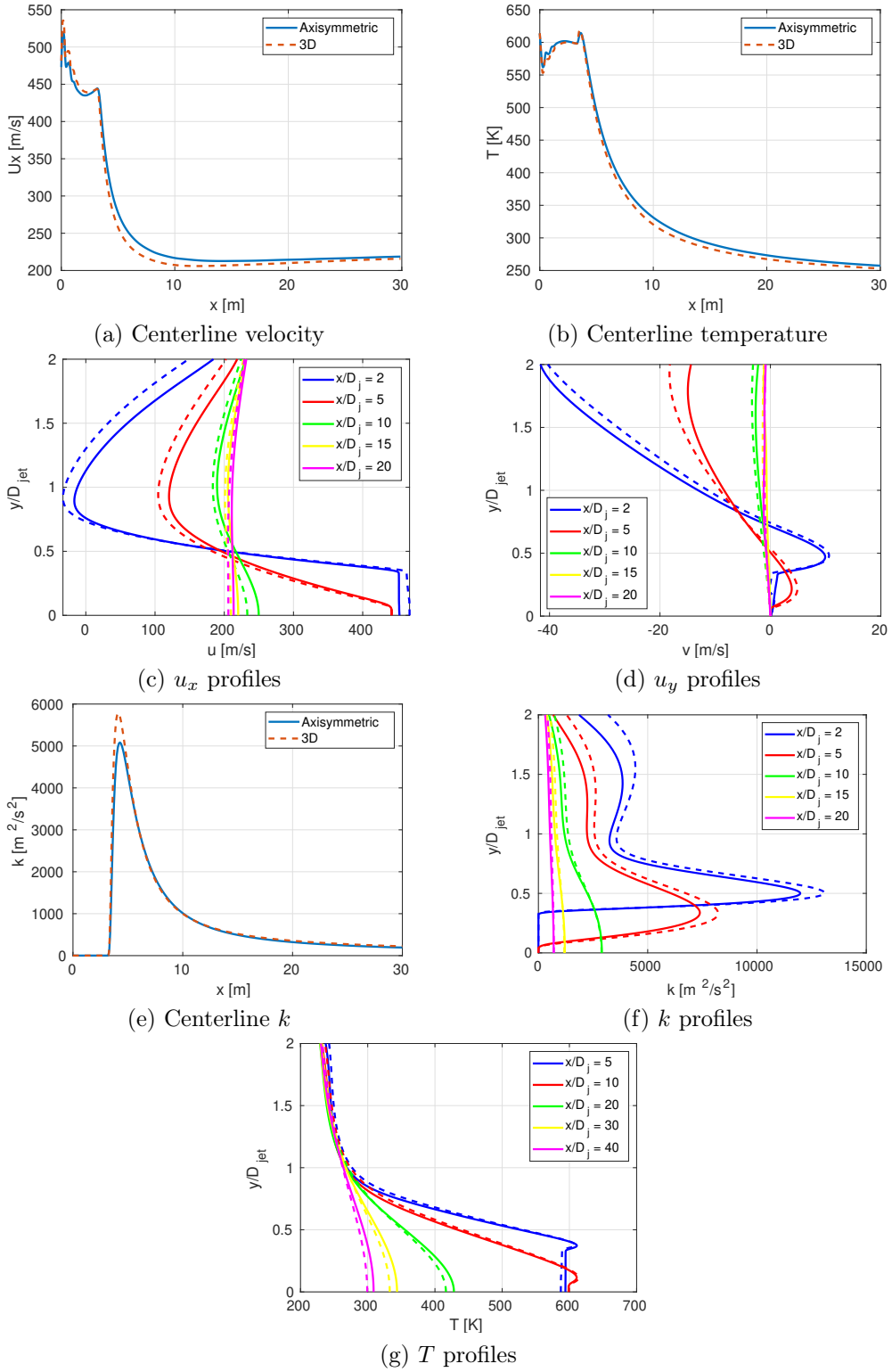
(a) Zoom potential core wall case - Group 1



(b) Zoom potential core wall case - Group 2

Figure 6.23: Zoom end of the potential core region 3D wall cases

6.3.3 Comparisons Axisymmetric-3D



(a) Centerline velocity

(b) Centerline temperature

(c) u_x profiles

(d) u_y profiles

(e) Centerline k

(f) k profiles

(g) T profiles

Figure 6.24: Comparisons between the 3D axisymmetric and the full 3D wall cases. 3D axisymmetric (Solid line -) full 3D (Dashed line - -)

Figure 6.13 shows the comparison of the results obtained with Mesh 1 (axisymmetric) and Mesh 2°(full 3D) for the wall case. As for the no-wall case, the axisymmetric

model tends to predict a lower decay of the centerline velocity. This is due to an higher prediction of the turbulent kinetic energy k of the 3D model with a consequent increase of the turbulent viscosity ν_T. About the length of the potential core it is the approximately the same in both cases (3.015 m for the axisymmetric case and 3.177 m for the full 3D case), while the potential core velocity is higher in the full 3D case (u_{max} 3D 535.8 m/s, u_{max} axisymmetric 514.7 m/s). This contradicts again the conclusion of [39]. Anyway, far from the nozzle exit, when the flow is fully developed and becomes self-similar the results of the two models agree well one to each other.

6.4 DES simulations

The last step of this work is the use of the $k-\omega$ SST DES turbulence model to simulate the nozzle flow in both the no-wall and wall cases. Since for DES simulations the grid refinement is very important, and as the mesh cells become smaller, more turbulent eddies can be captured, it has been decided to perform the simulations on the most refined grid of the two cases (Mesh 2° with 23,662,125 cells for the no wall case and 20,130,070 for the wall case). Due to the high computational cost of the simulations, the steady state solutions obtained with the RANS approach have been used as initial conditions for both the simulations. This is possible because the structure of the equations to solve is exactly the same in both models (see Sections 2.2.2 and 2.2.4) and there is no need to make changes in the fields obtained using the RANS approach. The only changes have been made in the *turbulenceProperties* and in the *fvSchemes* files. For DES type simulations in the *turbulenceProperties* file it has been changed the turbulence model from $kOmegaSST$ to $kOmegaSSTDES$ and it has been defined the filter width as the maximum cell dimension according to (2.66). In the *fvSchemes* to guarantee an higher order of accuracy, the time scheme has been changed from Euler to Crank-Nicholson, while to have good results in the energy transferred from the sub-grid scales to the resolved scales the divergence schemes of k and ω have been changed from a pure upwind scheme to the LUST scheme [43]. Finally, because the high computational cost and the strong dependency of the solution from the grid size, no grid convergence study has been performed.

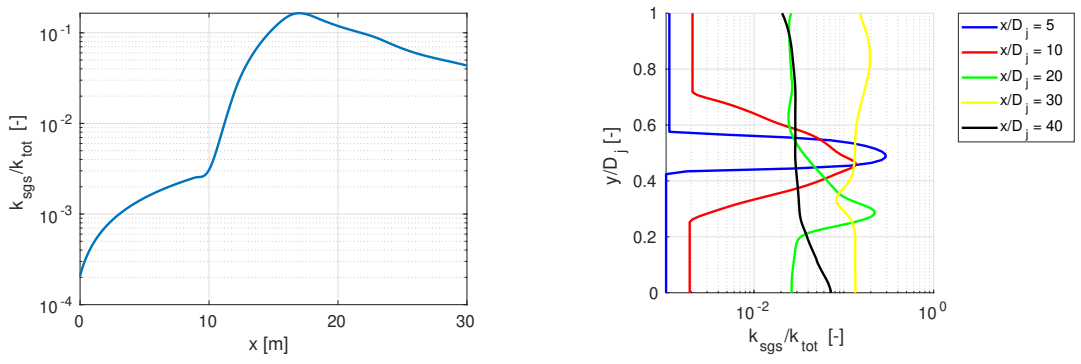
6.4.1 No wall case - DES approach

In the DES model the turbulent fluctuations are modeled, therefore it is not possible to reach a time steady solution but a statistically steady solution, which means that for a period time longer than the turbulent time scales, the average of the simulation quantities does not change. To have an estimation of when the simulation reached the statistically steady state a temperature probe has been inserted at a location six meters over the nozzle exit. When the temperature measured by the probe has started to measure a constant mean temperature, the statistics have been reset and restarted. Considering the velocity field, the statistics are computed as:

$$\langle \overline{u}_i \rangle = \frac{1}{N} \sum_{k=1}^{N} \overline{u}_i^{(k)} \tag{6.1}$$

$$\langle u_i' u_j' \rangle = \frac{1}{N} \sum_{k=1}^{N} \left(\overline{u}_i^{(k)} - \langle \overline{u}_i \rangle \right) \left(\overline{u}_j^{(k)} - \langle \overline{u}_j \rangle \right) \tag{6.2}$$

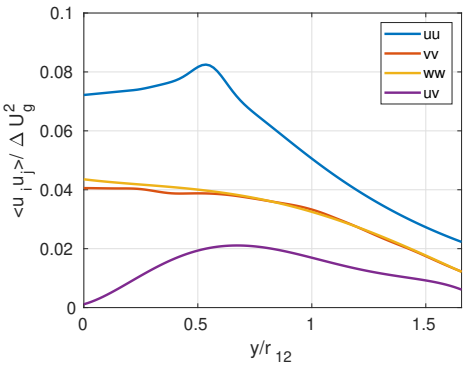
Equation (6.1) allows the computation of the average filtered velocity while (6.2) gives an estimation of the Reynolds stress tensor computed using the filtered quantities. It must be noticed that the statistics computed using (6.1) and (6.2) are different from the one used in the RANS formulation, indeed in the RANS approach the velocity is decomposed using (2.7), where u_i' is the velocity fluctuations from the mean velocity $\langle u_i \rangle$ and not from the filtered average velocity $\langle \overline{u}_i \rangle$. In the same way the quantity u_i' used in the LES velocity decomposition (2.44) is different from the square root of the tensor's diagonal given by (6.2), because in (2.44) u_i' is the velocity contribution of the sub-grid scales. Using the same relations the statistics can be computed also for the pressure and the temperature.
For the no-wall case the simulation has been run for 1.2s starting from the RANS steady state, corresponding to approximately 953 convective times ($T_{conv} = D_j/U_j$), while the statistics have been recorded for 0.74s corresponding to 588 convective times. To understand if the LES part of the model is used in the majority of the domain, Figure(6.25) shows on a log-scale the ratio of the turbulent kinetic energy of the RANS part of the model (it is equivalent to the sub-grid scales turbulent kinetic energy) and the total turbulent kinetic energy, along the centerline and at different locations over the nozzle exit. This last quantity is computed as the sum of the RANS turbulent kinetic energy and the LES turbulent kinetic energy, which is obtained multiplying by one-half the trace of the tensor given by (6.2). From Figure (6.25) it is well evident how the simulation is well resolved and that the most critical zone is the nozzle lipline in the first 20 diameters over the nozzle exit, where at x/D_j the sub-grid scales turbulent kinetic energy is approximately 30 % of the total. The validity of the computed statistics is checked computing the Reynolds stresses profiles and comparing them with the results of Hussein et al. available in [34] (see Figure 6.26b). These results are for a jet into quiescent air but can be used for a jet in coflow re-scaling $< u_i u_j >$ by the centerline excess velocity $\Delta U_g = U_g - U_a$ and using r_{12} as the location where $U - U_a = \frac{\Delta U_g}{2}$ that for $x/D_j = 30$ is 0.368 m. These results are valid in the jet's self similar region and in this case are computed at a location 30 D_j over the nozzle exit. For the same location Figure 6.26c shows the contribution to the total turbulent kinetic energy of the RANS and LES model parts.



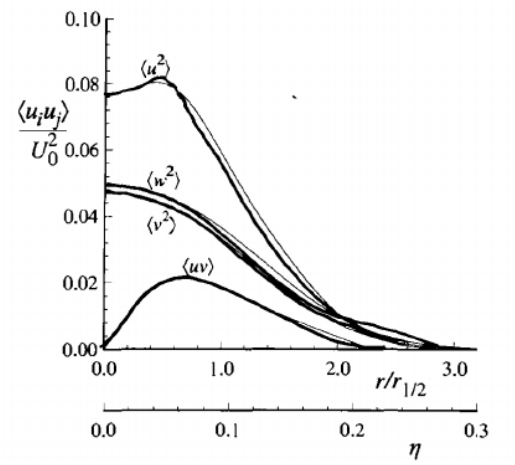
(a) Sgs centerline k contribution

(b) Sgs profiles k contribution

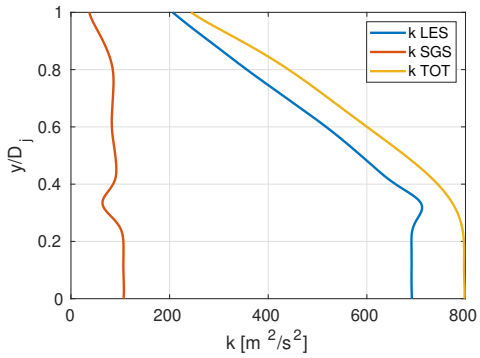
Figure 6.25: Sgs contributions to k no-wall case



(a) Reynolds stresses



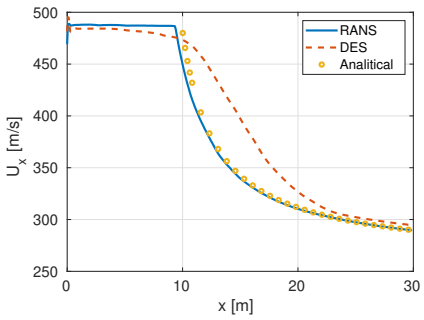
(b) Hussein et al.results



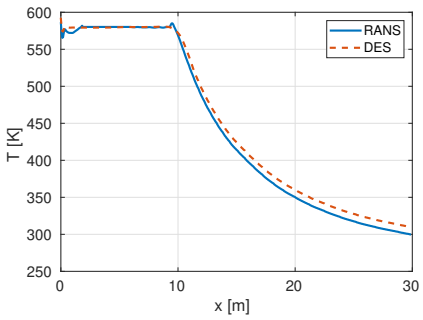
(c) Turbulent kinetic energy balance

Figure 6.26: Reynolds stresses comparisons no-wall case

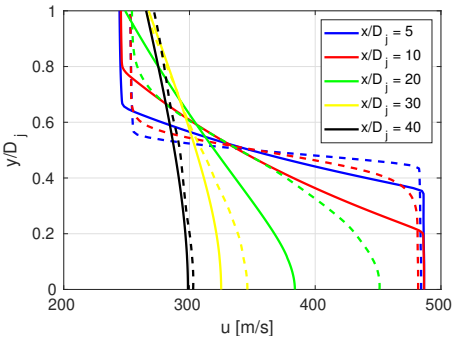
The results between 6.26a and 6.26b are in very good agreement, with the simulation data that have lower values compared to experimental ones. This is totally acceptable considering that the performed simulation uses a DES model and that the experimental data are for a jet in quiescent air. Figure 6.27 shows the simulation comparisons between the RANS and the average quantities of the DES model. Globally, the jet modeled using the DES model shows a less diffusive behavior compared to the RANS case. Both temperature and velocity scales slowly along the jet centerline, even though in the far field both the quantities scales in the same way, as it shown in Figure 6.28 where they are plotted on a log scale. The lower velocity decay of the DES model can be explained considering the turbulent kinetic energy profiles that especially few diameters over the nozzle exit have a lower peak compared to the RANS case.



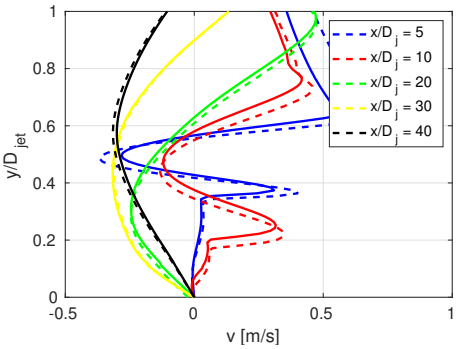
(a) Centerline velocity



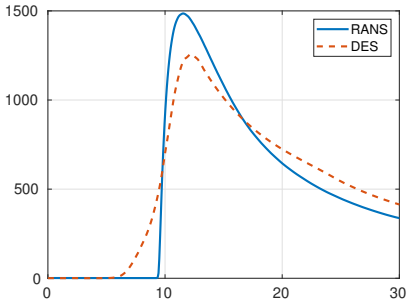
(b) Centerline temperature



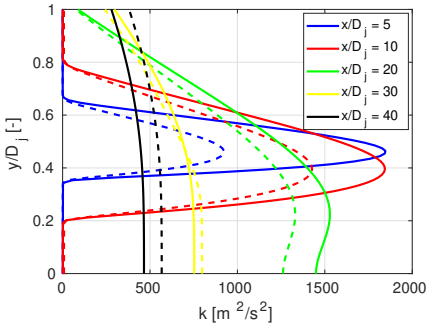
(c) u_x profiles



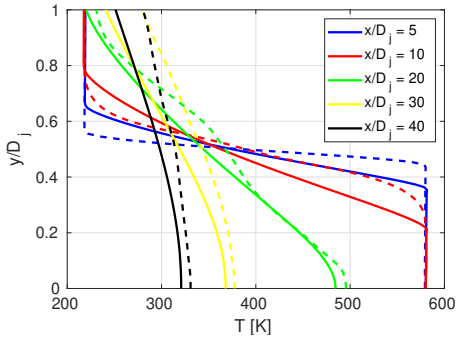
(d) u_y profiles



(e) Centerline k



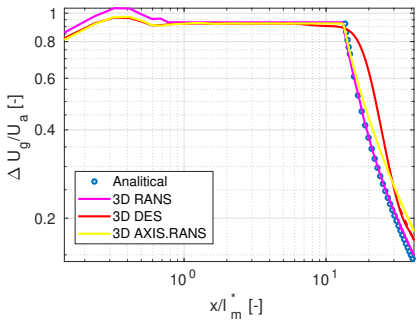
(f) k profiles



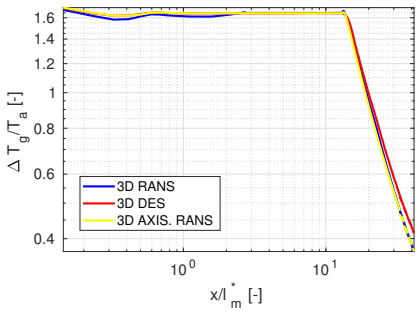
(g) T profiles

Figure 6.27: Comparisons between the DES and RANS model for the no-wall case on Mesh 2° .Solid line (-) RANS, Dashed line (- -) DES

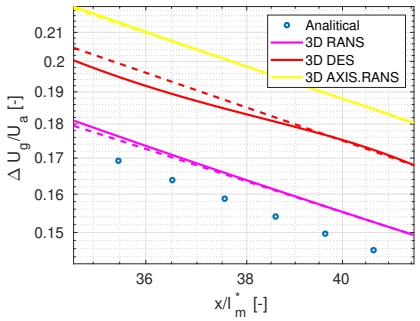
Figure 6.28 gives the decays of the ceterlines temperature and velocity on a logarithmic scale along the whole length (Figures 6.28a - 6.28b) and in the last 4 m of the domain (Figures 6.28d -6.28). The logarithmic scale is used in order to better compare the simulation results with the experimental data provided by [38] (Figure 6.29). Since at the end of the domain the flow is not completely in the strongly advected region ($x/l_m^* \geq 60$), it has been tried to estimate a power law decay at $x = 30m$ similar to the ones provided by (5.9) and (5.10). Table 6.5 provide the power laws for the velocity and temperature centerline spreading. In the power law fitting l_m^* has been evaluated as a mean of the three outlet velocities ($U_o = 485.1$ m/s) in order to have the same scaling quantity for all the three cases, while as for the notation used in Section 5.2.4, T_a represents the coflow temperature and ΔT_g the centerline temperature excess.



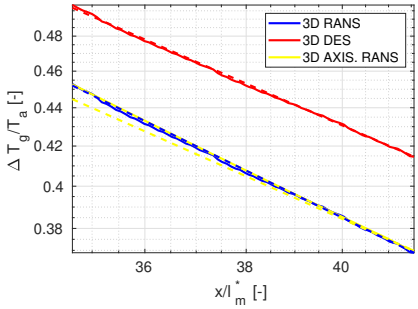
(a) Centerline velocity



(b) Centerline temperature



(c) Centerline velocity last 5m



(d) Centerline temperature last 5m

Figure 6.28: Centerline velocity and temperature decays for the no-wall case. Solid line (-) Simulation, Dashed line (- -) Power law fitting

	Centerline decays	
	Velocity	Temperature
2D RANS	$\frac{\Delta U_g}{U_a} = 5.15 \left(\frac{x}{l_m^*}\right)^{-0.90}$	$\frac{\Delta T_g}{T_a} = 16.03 \left(\frac{x}{l_m^*}\right)^{-1.01}$
3D RANS	$\frac{\Delta U_g}{U_a} = 6.43 \left(\frac{x}{l_m^*}\right)^{-1.01}$	$\frac{\Delta T_g}{T_a} = 23.55 \left(\frac{x}{l_m^*}\right)^{-1.12}$
3D DES	$\frac{\Delta U_g}{U_a} = 4.69 \left(\frac{x}{l_m^*}\right)^{-0.89}$	$\frac{\Delta T_g}{T_a} = 16.79 \left(\frac{x}{l_m^*}\right)^{-0.99}$

Table 6.5: Centerline velocity and temperature decay for the no-wall case

Comparing the centerline velocity decay it is interesting to notice how both the

exponent of the power law and the constant that multiplies the adimensional axial coordinate, have values between the ones given by (5.9) and (5.10). This again shows how for $x = 30m$ the jet is between the strongly advected and the weakly advected region and how in the last meters of the domain the experimental data are satisfied by the three simulations. However, the domain length between $15 < x/l_m^* < 26$, the DES case is in poor agreement with experimental data. Here the the DES centerline velocity decays too slow, suggesting a not optimal mesh refinement in this zone.

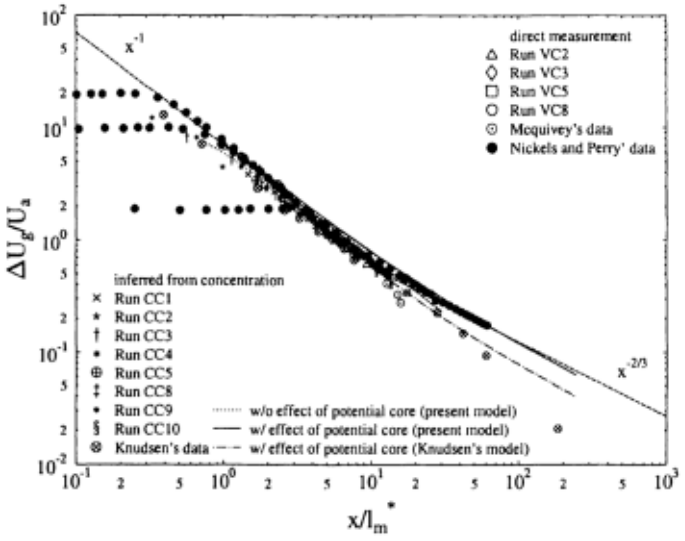


Figure 6.29: Experimental data

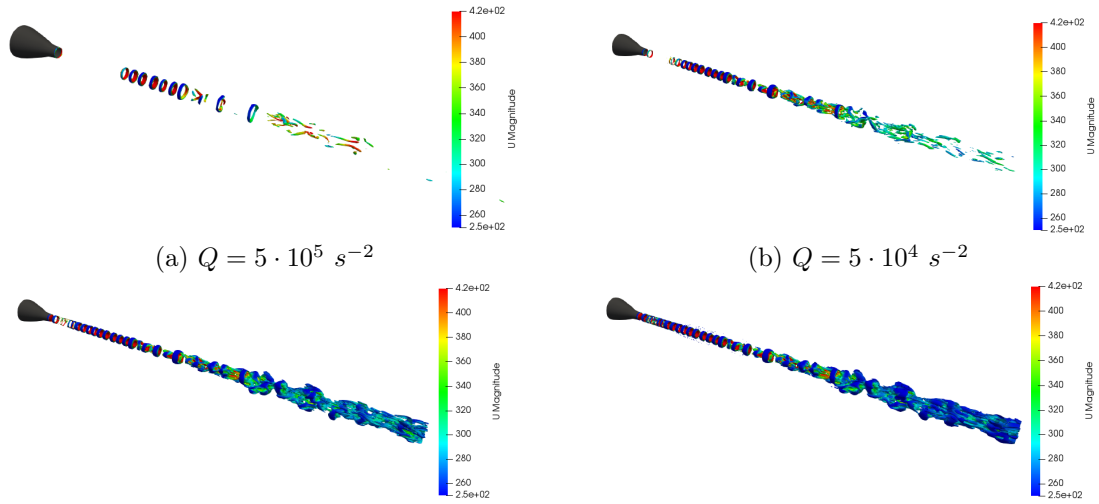
Since the main of the DES simulation is to identify vortex structures Figure 6.30 shows the isosurfaces of different Q criterion values colored with the velocity magnitude. The *Q-criterion* defines a vortex as a "connected fluid region with a positive second invariant of ∇u ".[44]. Considering the velocity gradient tensor $\overline{D} = \frac{\partial u_i}{\partial x_i}$, the second invariant is defined as:

$$Q = \frac{1}{2}\left(tr(\overline{D})^2 - tr(\overline{D}^2)\right) = \frac{1}{2}||\overline{\Omega}^2|| - ||\overline{S}^2|| \tag{6.3}$$

Where $S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the well know rate of strain tensor, while $\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)$ is the vorticity tensor. Therefore considering the definition given by (6.3), the Q criterion represents the local balance between shear strain rate and vorticity magnitude, defining vortices as areas where the vorticity magnitude is greater than the magnitude of rate-of-strain [45] .

As it is possible to see in Figure 6.30 the turbulent structures start to be well formed over the jet potential core length. Inside the potential core region, the isosurfaces have an annular shape, meaning that inside this region the turbulence has not started to arise and that the flow is still laminar. This explains why in the DES case the turbulent kinetic energy is lower in this part of the domain and why the potential core length is longer compared to the axisymmetric and 3D RANS cases, without a well defined transition between its end and the start of the centerline decay.

To check that the computed statistics have a physical meaning along all the domain, Figure 6.31 shows the instantaneous profiles of the centerline temperature

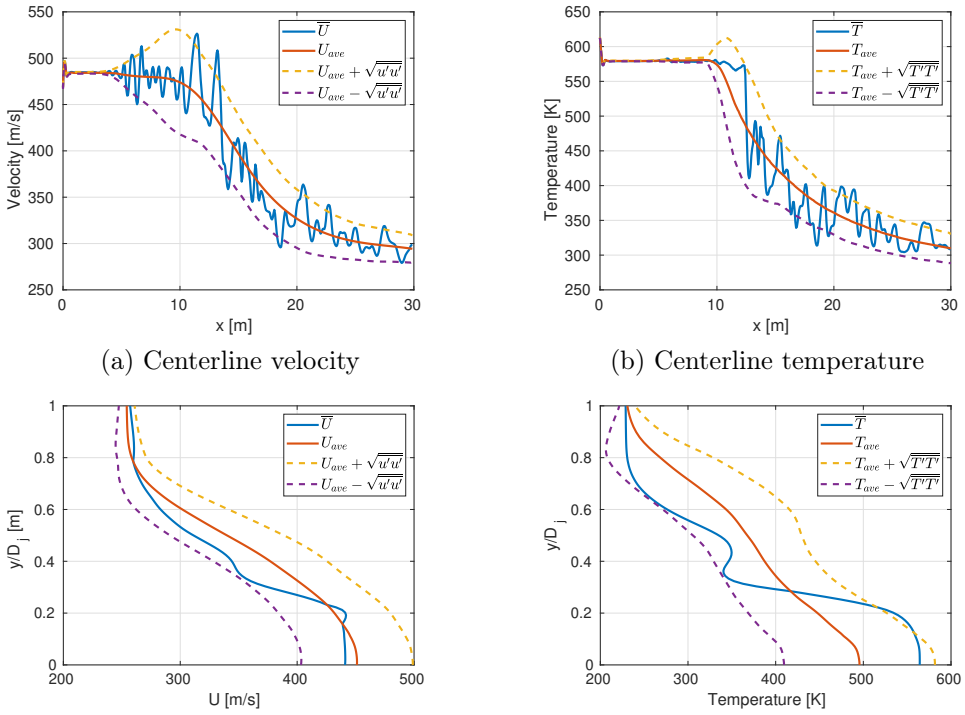


(a) $Q = 5 \cdot 10^5\ s^{-2}$ (b) $Q = 5 \cdot 10^4\ s^{-2}$

(c) $Q = 5 \cdot 10^3\ s^{-2}$ (d) $Q = 5 \cdot 10^2\ s^{-2}$

Figure 6.30: Q criterion isosurfaces $u \in [250, 420]\ m/s$

and velocity and at a location twenty diameters over the nozzle exit (this location is chosen for the no-wall case because at this point the flow is well out of the potential core region and the centerline turbulent kinetic energy is near its peak). They are plotted together with the mean profiles and the the mean profiles with added and subtracted the turbulent fluctuations. In this way it can be shown that the instantaneous profiles fall between the two limits given by the mean profiles $\pm$ the turbulent fluctuations, showing the validity of the simulation fluctuations.



(a) Centerline velocity (b) Centerline temperature

(c) u_x $x/D = 20$ (d) T $x/D = 20$

Figure 6.31: Profiles with turbulent fluctuations no-wall case

Finally, Figures 6.32-6.33 show the instantaneous and average contour plots of

temperature and velocity computed for the last time step of the simulation. The stochastic nature of turbulent solutions to the full Navier-Stokes is well evident in the DES model, where the instantaneous snapshots provide a real time-dependent view of the resolved scales, especially at the end of the domain where the jet starts to separate. To better show the different scales in the instant snapshot of the velocity magnitude, Figure 6.32 has been re-scaled to the the values in the interval [250 500] m/s. Looking at the average contour plot of the DES model velocity (Figure 6.33) and at the one of the RANS case (Figure 6.6), it is well evident how the potential core predicted by the DES case is longer compared to the one of the RANS case, as well as the jet width, this again confirms the lower turbulent kinetic energy and mixing predicted by DES model.

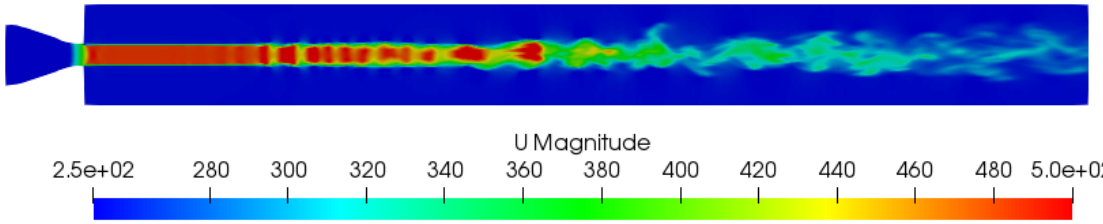


Figure 6.32: Istantaneous U snapshot no-wall case

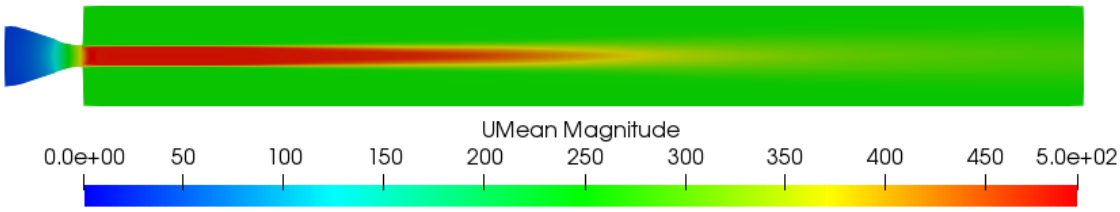


Figure 6.33: U mean snapshot no-wall case

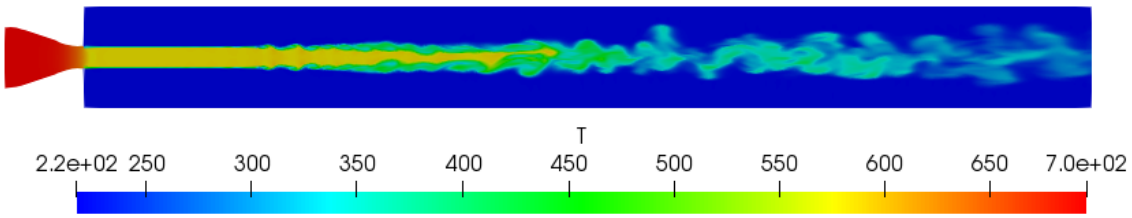


Figure 6.34: Istantaneous T snapshot no-wall case

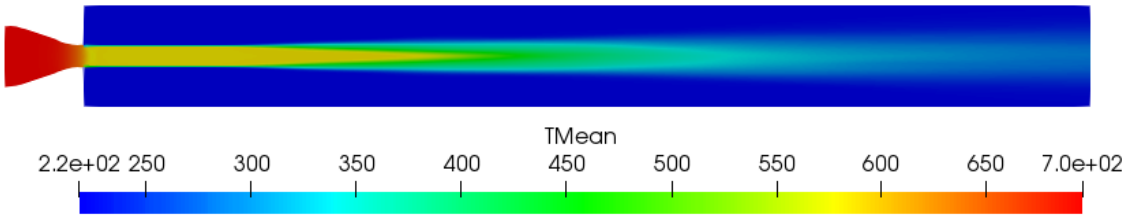
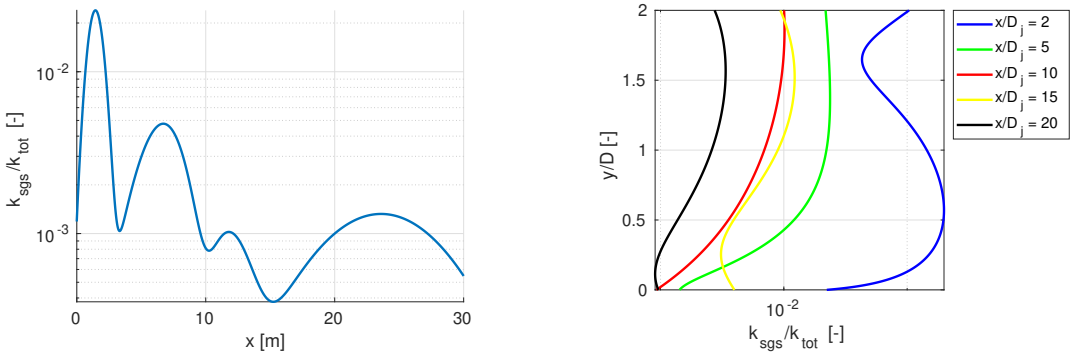


Figure 6.35: T mean snapshot no-wall case

6.4.2 Wall case - DES approach

Here are reported the results for wall case following the same order of the no-wall case. In this last case, the simulation has been run for 1.15s (913 convective times) and the statistics have been recorded for 0.6s (477 convective times).
To decide the time at which starting to record the statistics, it has been used the same approach of the no-wall case, with the only difference that the probe has been posed at a location 3 m over the nozzle exit, due to the shorter jet potential core length of this case. In the wall case even if the mesh is less refined, looking at Figure 6.36 it is possible to notice how the contribution of the SGS to the total turbulent kinetic energy is lower compared to the no-wall case. This can be explained taking in consideration the recirculation zone above the nozzle exit. In this region the flow detaches from the wall and has a lower velocity compared to the coflow and the flow exiting from the nozzle. Remembering that the turbulent scales are proportional to $Re_L^{9/4}$, the slower is the flow and less refined the mesh has to be in order to guarantee a good LES simulation. Another effect of the recirculation zone is that in this case the contribution to the turbulent kinetic energy of the SGS is mainly in the first five diameters over the nozzle exit. It is along this zone that the recirculation zone develops, increasing turbulence and mixing that kill the jet exiting from the nozzle. This explain why in this zone the LES part of the model can not resolve all the turbulent scales and need the help of the RANS part.



(a) Sgs centerline k contribution

(b) Sgs profiles k contribution

Figure 6.36: Sgs contributions to k wall case

Figure 6.37 shows the velocity streamlines around the recirculation zone computed at the last timestep using the DES model. It is well evident how the symmetry shown in the RANS model (see Figure 6.14) completely disappears and the streamlines have a chaotic behavior that changes every time step underlining the stochastic behavior of the Navier-Stokes equations.

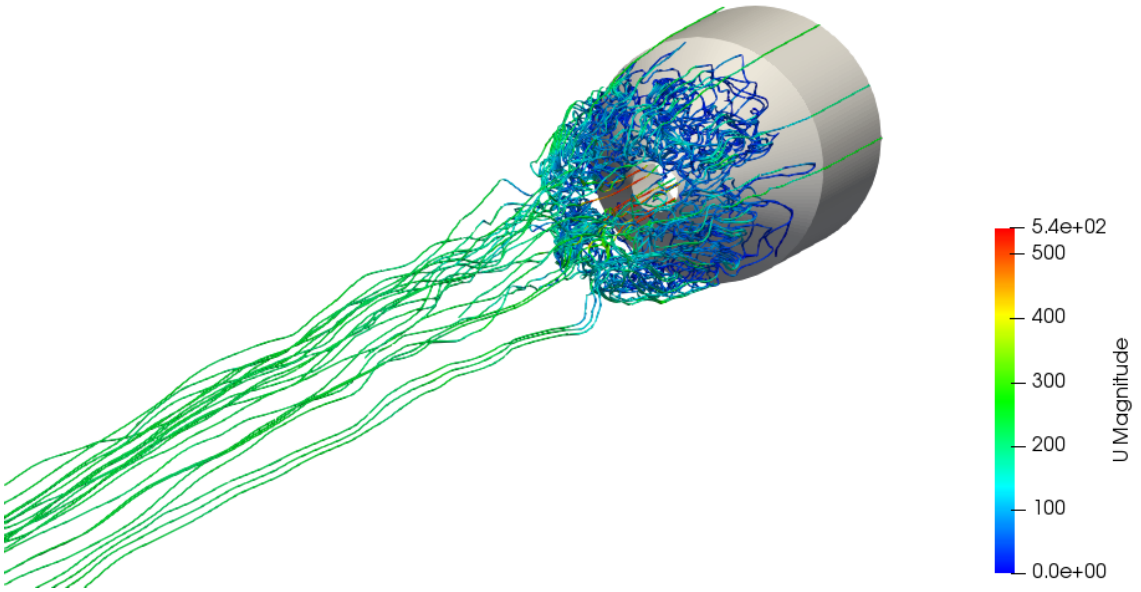
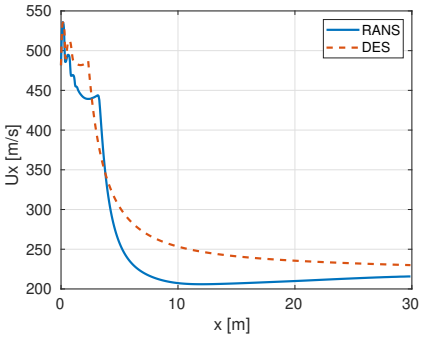
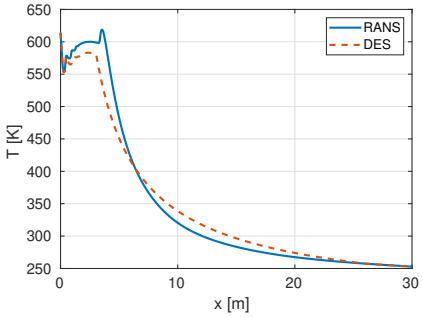


Figure 6.37: 3D streamlines wall case - DES approach

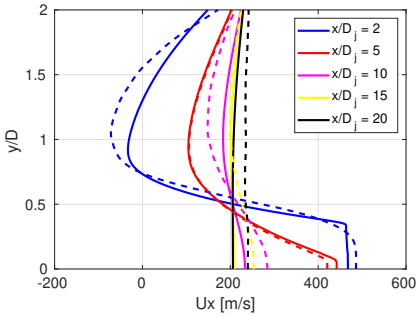
Figure 6.38 shows the comparisons between the RANS and the DES models in the wall case. As for the no-wall case, in the DES model the centerline temperature and the centerline velocity have a lower decay compared to the RANS case, this can be again explained looking at the turbulent kinetic energy that is lower in the DES case especially in the first diameters over the nozzle exit. However, it can be noticed that at the end of the domain in the DES case, the centerline velocity continues to decay, while in the RANS it starts to slowly increase. Figure 6.39 shows the average centerline Mach number and the average lipline pressure and velocity, for the three wall cases analyzed since now. The DES case is the one that predicts the highest velocity peaks and the longest estimated recirculation zone, while the shortest recirculation zone is predicted by the axisymmetric case. This can be explained considering that in the axisymmetric approximation, it is not considered the turbulence around the angular direction and this tends to underestimate the recirculation phenomena. The higher velocity and the lower pressure above the nozzle lipline have their effect on the predicted potential core length of the DES case that has a shorter length but an higher Mach compared to the axisymmetric and 3D RANS cases, indeed the lower pressure promotes the increase of the velocity out of the nozzle exit, but at the same time it calls back fluid from the nozzle flow reducing the potential core length. The underestimation of the of the recirculation of the two RANS cases compared to the DES case, can be attributed to the well-known deficiency of two-equation models regarding the over-prediction of the turbulent kinetic energy in regions with large normal strain (flow region with strong acceleration or deceleration) as well as the poor prediction of three-dimensional effects in flows with strong separation as in this case.



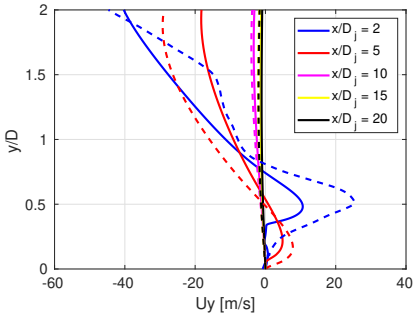
(a) Centerline velocity



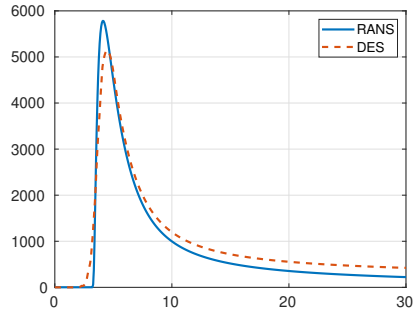
(b) Centerline temperature



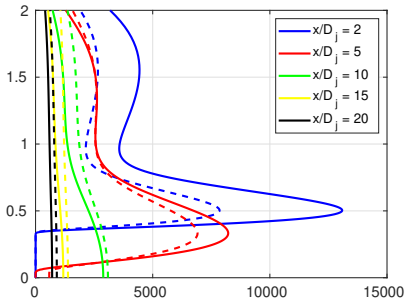
(c) u_x profiles



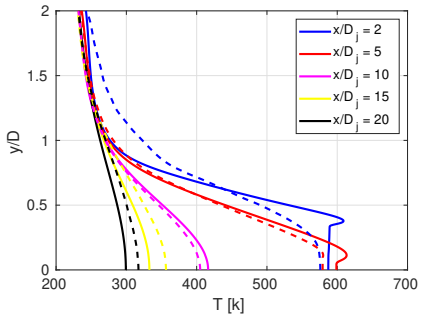
(d) u_y profiles



(e) Centerline k

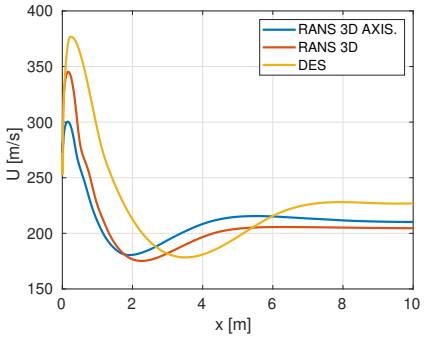


(f) k profiles

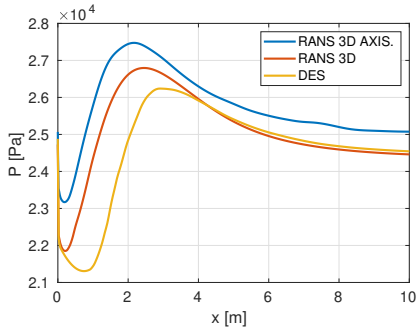


(g) T profiles

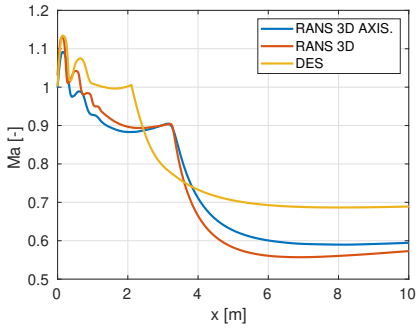
Figure 6.38: Comparisons between the DES and RANS model for the wall case on Mesh 2° Solid line (-) RANS, Dashed line (- -) DES



(a) Lipline velocity



(b) Lipline pressure



(c) Centerline Mach

Figure 6.39: Mach number and lipline velocity and pressure for the three wall cases

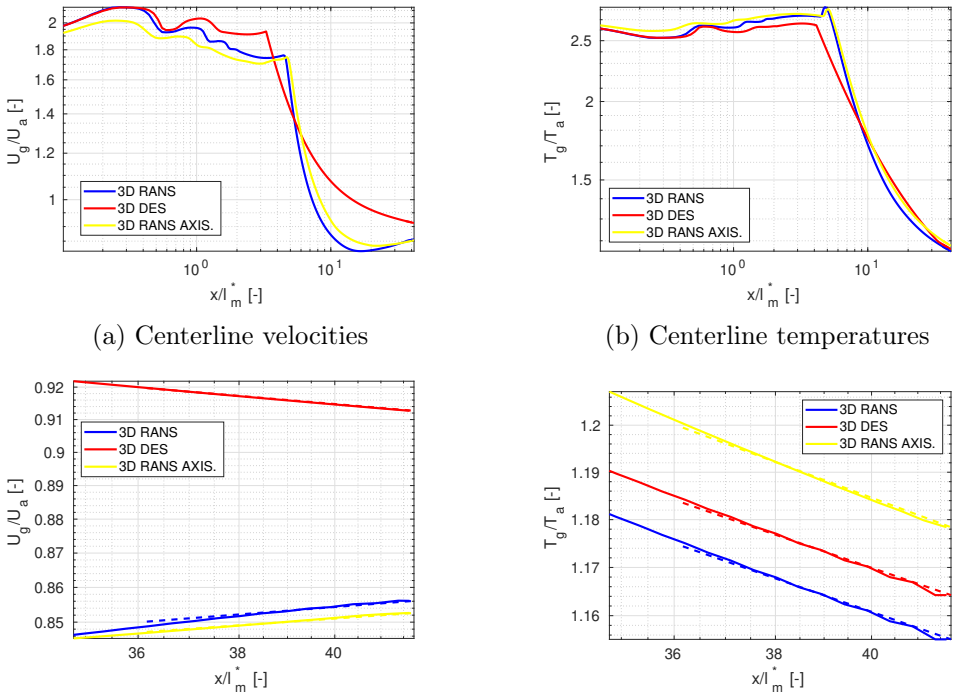
This is due to the Boussinesq approximation that assumes alignment of the Reynolds stress anisotropy and the strain, that leads to a systematic overestimation of the turbulent kinetic energy production when turbulence an strain rate are not aligned. This leads to high values of the eddy viscosity that may result in a damping of the flow oscillations inside the recirculation zone [46]. However, it must be said that the $k-\omega$ SST part of the DES model is used in the wall region of the nozzle duct, therefore even the DES can not be considered free of the turbulent kinetic energy overestimation. Table 6.6 summarize the estimated length for the recirculation zone in the three simulated cases.

	Estimated recirculation zone length
AXIS. RANS	1.73 m
3D RANS	2.10 m
DES	3.25 m

Table 6.6: Estimated recirculation zone length for the three wall cases

As for the no wall case, it has been tried to estimate a scaling law for the centerline temperature and velocity at the end of the domain. However, because the analytical model of the coflow jet is not valid anymore, the power law fitting has been computed as $\frac{U_g}{U_a} = A\left(\frac{x}{l_{m*}}\right)^n$ for the velocity and $\frac{T_g}{T_a} = A\left(\frac{x}{l^*_m}\right)^n$ for the temperature. Figure 6.40 provides on a logaritmic scale the centerline velocity and temperature decay along the whole length (Figures 6.40a and 6.40b) and in the last 5 m of the domain (Figures 6.40c and 6.40d, while Table 6.7 gives the power law fitting for each of the analyzed cases. The temperature scaling is very similar in all the three

simulation, while for the velocity scaling,the RANS cases show a positive trend and the DES case a negative trend with the centerline velocity that continues to reduce. Figure 6.41 shows the Q criterion for the same values of the no-wall case. What is s immediately evident, is that for the wall cases with the same value of Q criterion it is possible to identify an higher number of vortex structures. These structures are mainly due to the recirculation zone over the nozzle exit and are characterized by a velocity magnitude of the same order of the coflow value. However, approaching the end of the domain especially for low values of the Q-criterion, they have a strange elongated shape showing that here the model is inside the grey zone, where the length scale of the RANS model is comparable to the length scale of the LES model (See Section 2.2.4).



(a) Centerline velocities

(b) Centerline temperatures

(c) Centerline velocities last 5m

(d) Centerline temperatures last 5m

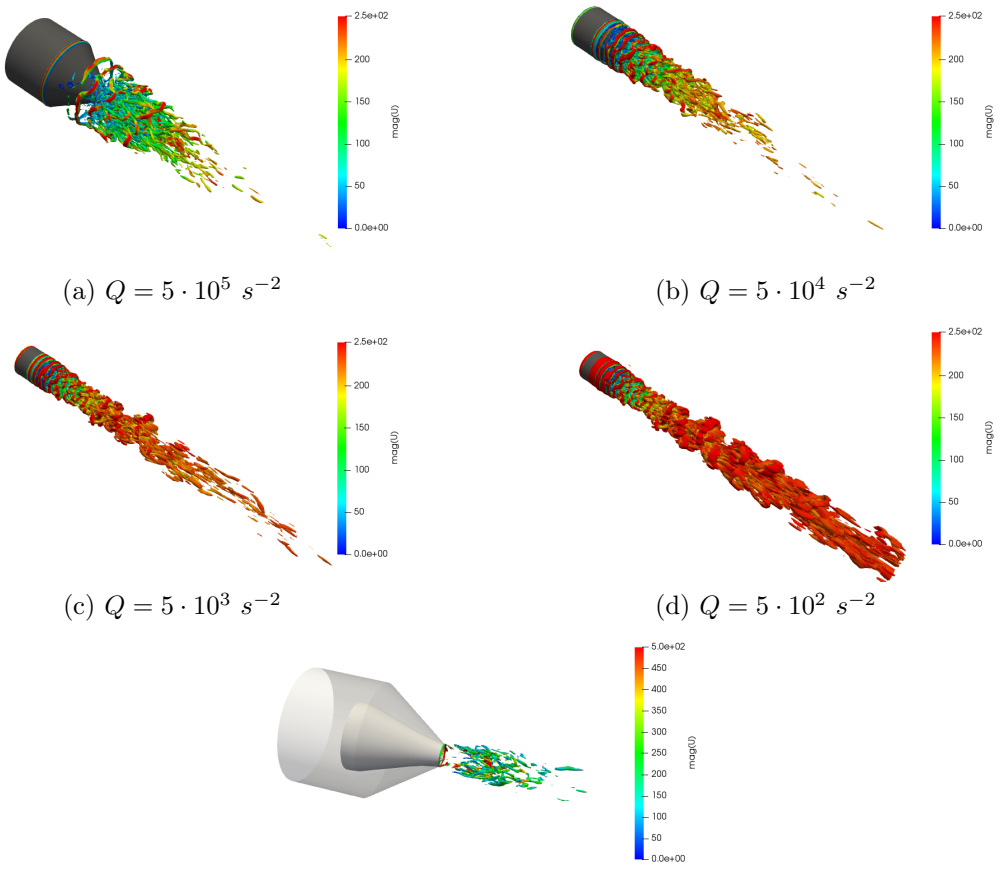
Figure 6.40: Centerline velocity and tempeature decays for the wall case at the end of the domain. Solid line (-) Simulation, Dashed line (- -) Power law fitting

	Centerline decays	
	Velocity	Temperature
2D RANS	$\frac{U_g}{U_a} = 0.72 \left(\frac{x}{l_m^*}\right)^{0.04}$	$\frac{T_g}{T_a} = 1.87 \left(\frac{x}{l_m^*}\right)^{-0.12}$
3D RANS	$\frac{U_g}{U_a} = 0.71 \left(\frac{x}{l_m^*}\right)^{0.05}$	$\frac{T_g}{T_a} = 1.77 \left(\frac{x}{l_m^*}\right)^{-0.12}$
3D DES	$\frac{U_g}{U_a} = 1.11 \left(\frac{x}{l_m^*}\right)^{-0.05}$	$\frac{T_g}{T_a} = 1.79 \left(\frac{x}{l_m^*}\right)^{-0.12}$

Table 6.7: Centerline velocity and temperature decay for the no-wall case

This suggest that it is necessary a mesh refinement in this zone to obtain a more accurate results and this could be an explanation of why the centerline velocity in the

DES case decays slower compared to the RANS case. To identify vortex structures near the nozzle exit it is necessary increase the Q-criterion reaching a value of $6 \cdot 10^6$ s^{-2}. For this value the turbulent structures have no annular shape as in the no wall case, showing how the recirculation effect created by the nozzle duct strongly increases the arise of turbulence and flow instability. This also explains why for the wall case, the turbulent kinetic energy is more than five times higher compared to the no-wall case. Figures 6.42-6.45 show the instantaneous and the mean contour plots for the velocity and temperature, in the instantaneous snapshots, to better show the turbulent structures at the end of the domain, the velocity magnitude has been rescaled to $u \in [0, 300]$ m/s and the temperature to the interval $T \in [210, 400]$ K. Finally, as for the no wall case, to check the physical meaning of the computed statistics, Figure 6.46 shows the instantaneous profiles and the mean profiles $\pm$ the turbulent fluctuations, for the centerline and lipline temperature and velocity, together with the radial profiles at a location 5 diameters downstream of the nozzle exit. This location correspond approximately to the end of the potential where the centerline turbulent kinetic energy starts rapidly to increase.



(a) $Q = 5 \cdot 10^5$ s^{-2}

(b) $Q = 5 \cdot 10^4$ s^{-2}

(c) $Q = 5 \cdot 10^3$ s^{-2}

(d) $Q = 5 \cdot 10^2$ s^{-2}

(e) $Q = 6 \cdot 10^6$ s^{-2}

Figure 6.41: Q criterion isosurfaces $u \in [250, 420]$ m/s for $Q \in [5 \cdot 10^2, 5 \cdot 10^5]$ s^{-2} and $u \in [0, 500]$ m/s for $Q = 6 \cdot 10^6$ s^{-2}

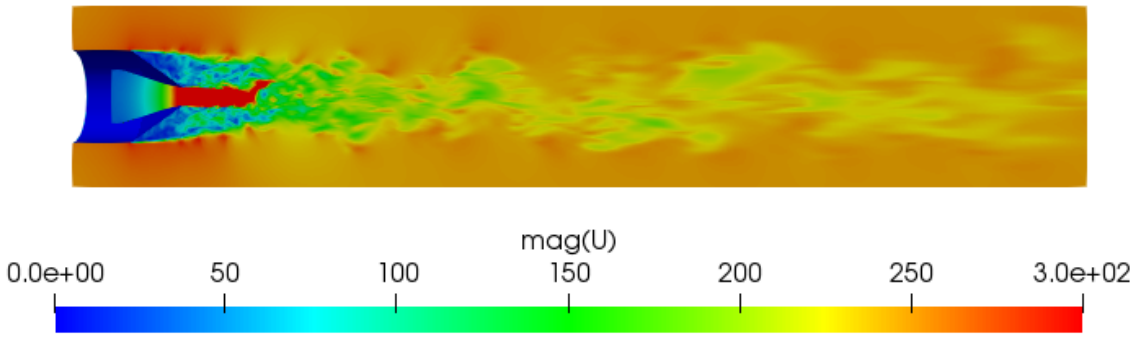


Figure 6.42: Istantaneous U snapshot wall case

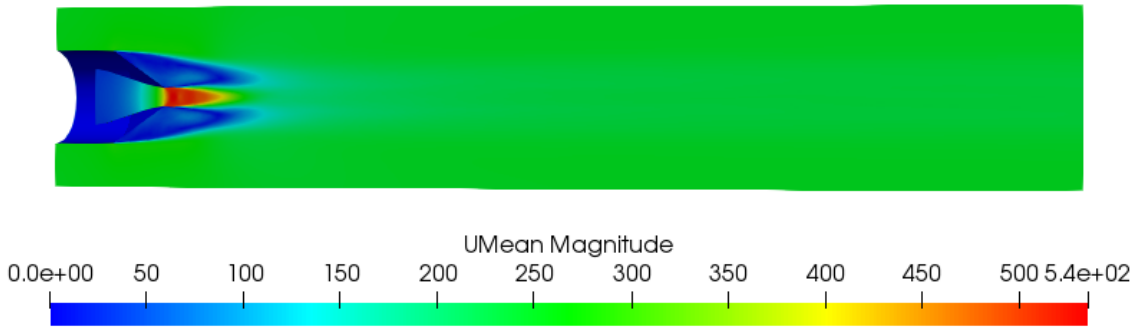


Figure 6.43: U mean snapshot wall case

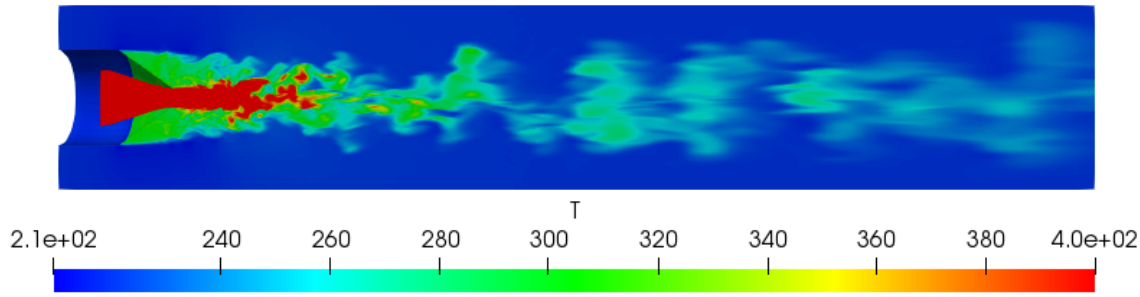


Figure 6.44: Istantaneous T snapshot wall case

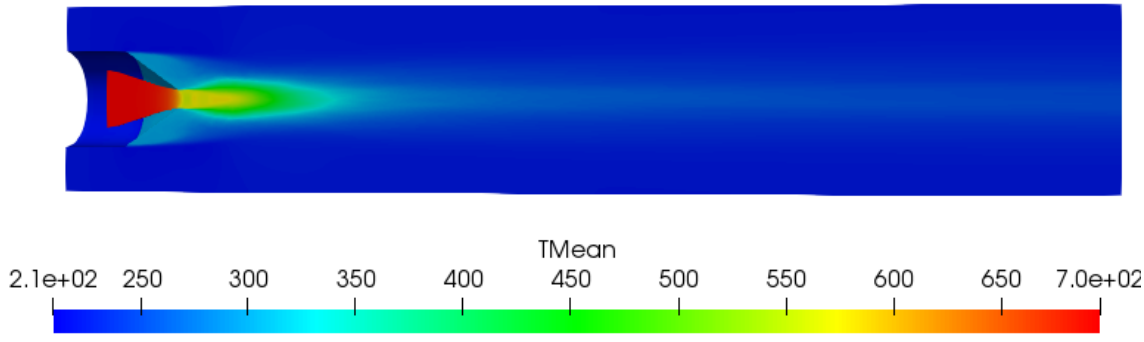
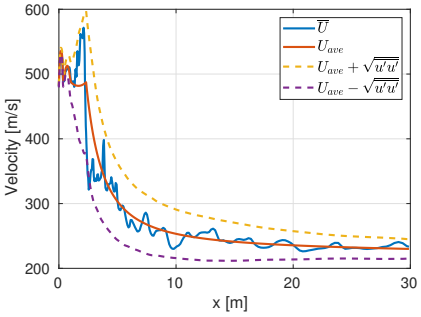
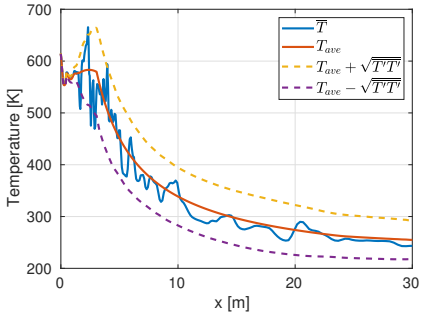


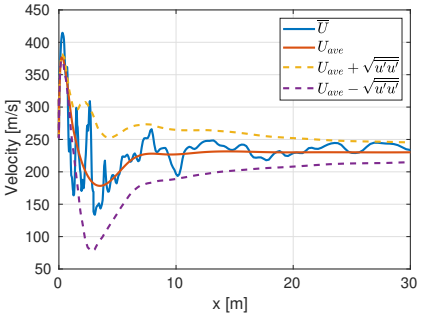
Figure 6.45: T mean snapshot wall case



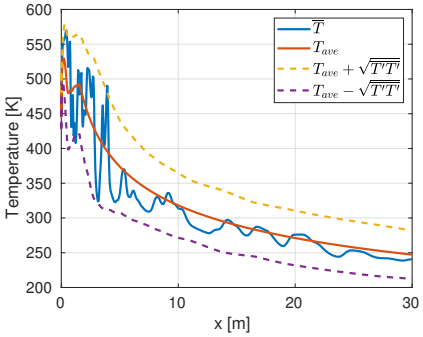
(a) Centerline velocity



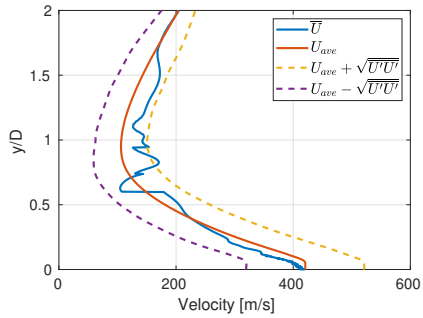
(b) Centerline temperature



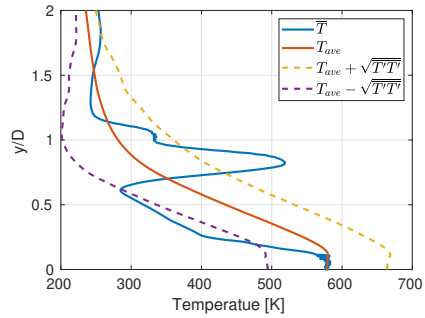
(c) Lipline velocity



(d) Lipline temperature



(e) U_x $x/D = 5$



(f) T $x/D = 5$

Figure 6.46: Profiles with turbulent fluctuations wall case

Chapter 7

Conclusions and further work

The main aim of this thesis work is the simulation of the Jet Regime in the contrail formation. As a preliminary study in the UIC project "High-performance computing and data-driven modeling of aircraft contrails", the exhaust flow exiting from a CFM-56 engine is simulated without considering soot particles that can act as nucleation points for the sublimation of the atmospheric water vapour. The geometry used for the drain nozzle of the CFM-56 engine is the one of the NASA ARN-2 nozzle, which has been rescaled to take in to account the exit diameter of 0.610 m of the CFM-56 engine. The simulations are performed with OpenFOAM and the set-up of the simulation parameters using the $k-\omega$ SST turbulence model has been carried on the NASA near sonic jet validation case [1]. The simulations are performed considering two type of geometries, an axisymmetric geometry and a full 3D geometry, both considering and not considering the wall effect of the duct surrounding the nozzle exit. As a final step in the work to show the turbulent structures and the vortex formation in the 3D case, the hybrid model $k-\omega$ DES is used and the obtained results are compared with the previous simulations. For the no-wall cases the results are compared with the analytical model and the experimental results for the centerline velocity spreading available in [38]. All the results are in well agreement with the analytical and experimental data , especially at the end of the domain where the jet flow is completely separated. In this region the jet is between the strongly advected and the weakly advected region and it scales in a way that is between the asymptotic relations given by (5.9) and (5.10). However, the DES model predicts a slower centerline decay between 10 and 20 m over the nozzle exit, this can be caused by a not sufficient grid refinement in this zone. The DES model allows also to show that without adding turbulence enhancing, the jet's turbulence structures begin to form after 15 m from the nozzle exit, explaining why the DES case predict a lower centerline velocity decay. The lower centerline decay it is also evident for the temperature, even if for this quantity there is a lower gap between the three cases. The physics of the problem completely change taking in consideration the wall effect of the duct over the nozzle that creates a recirculation zone for the coflow in this region of the domain. The lower pressure of the recirculation zone calls back fluid from the nozzle exit reducing the potential core length and enhancing the scaling of the centerline velocity and temperature. The length of the recirculation zone is estimated considering the lipline velocity profile of the nozzle, where the upper and lower peaks denote respectively the pressure drop due to the detachment of the coflow and the pressure peak due to the coflow reattachment.

The lower length of the recirculation zone is predicted by the 3D-axisymmetric case, this can be explained because in this case it is not considered the angular velocity of the flow due to the axisymmetric conditions and this may reduce the mixing between the detached coflow and the main flow exiting from the nozzle. The longer recirculation zone is instead predicted by the DES model, that consequently predicts a shorter potential core length. The underestimation of the recirculation zone by the axisymmetric and 3D RANS cases can be explained considering the Boussineq approximation that assumes alignment of the Reynolds stress anisotropy and the strain, that leads to a systematic overestimation of the turbulent kinetic energy production when turbulence and strain rate are not aligned. This leads to high values of the eddy viscosity that may result in a damping of the flow oscillations inside the recirculation zone. The effect of the recirculation zone strongly increases the arise of the turbulence compared to the no-wall case, indeed for the same Q-criterion the wall case shows a lot more turbulent structures that start from the inclined part of the nozzle duct. Morover, to identify the turbulent structures directly related to the flow exiting from the nozzle it is necessary to consider a value of the Q-criterion at least one order of magnitude grater of the no-wall case. Globally, the DES model predicts a lower turbulent kinetic energy, this explains why for this case the velocity and the temperature scale slowly compared to the axisymmetric and 3D RANS cases. The lower turbulent kinetic energy compared to results of the $k-\omega$ SST model, can be also seen in the experimental results of [31] and [32] for the NASA near sonic validation case, this confirms that the $k-\omega$ SST model tends to over predict this quantity giving the flow a more diffusive character. However, it must be remembered that in the $k-\omega$ SST DES turbulence model the turbulent kinetic-energy of the sub grid scales is computed using the RANS approach of the $k-\omega$ SST, therefore neither this hybrid model can be considered free of the turbulent kinetic energy overestimation. Finally, it must be said that these are numerical simulation and they always need an experiment to be correctly validated, this is particularly true for the wall case simulations where there are no available analytical models and experimental results, which are very difficult to obtain considering the high velocities involved in the flow. As stated an the beginning of this section, this is a preliminary work to model the Jet phase in the contrail formation, therefore the future development for this work will be the insertion of soot particles, that will be tracked in the flow using a Lagrangian approach and the implementation of a suitable thermophysical model to take into account the sublimation of the atmospheric water vapor on them. Moreover, if it will be possible a further mesh improving without increasing the computational cost,using for example a commercial software, it will be able to set-up a pure LES model in order to have a more realistic and reliable simulation.

Appendix A

Compressible LES equations

As for the averaging operation, for the derivation of the compressible LES equations it is useful to introduce the Favre trasformation:

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\overline{\rho}} \tag{A.1}$$

Applying (A.1) to (2.1), (2.2), (2.3) and (2.4) leads to:

$$\frac{\partial \overline{\rho}}{\partial t} + \overline{\frac{\partial}{\partial x_j}\left(\rho u_j\right)} = 0 \tag{A.2}$$

$$\frac{\left(\overline{\rho}\tilde{u_i}\right)}{\partial t} + \overline{\frac{\partial}{\partial x_j}\left(\rho u_i u_j\right)} = -\overline{\frac{\partial p}{\partial x_i}} + \overline{\frac{\partial \sigma_{ij}}{\partial x_j}} + \overline{\rho}\tilde{f}_i \tag{A.3}$$

$$\frac{\partial\left(\overline{\rho}\widetilde{E}\right)}{\partial t} + \overline{\frac{\partial\left(\rho u_j E\right)}{\partial x_j}} = -\overline{\frac{\partial p u_j}{\partial x_j}} + \overline{\frac{\partial\left(u_i \sigma_{ij}\right)}{\partial x_j}} - \overline{\frac{\partial q_j}{\partial x_j}} + \overline{S_E} \tag{A.4}$$

$$\overline{p} = \overline{\rho} R \tilde{T} \text{and } \tilde{i} = c_v \tilde{T} \tag{A.5}$$

Making the assumption that the spatial filter commutes with the derivatives operators the equations then becomes:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u_j}\right) = 0 \tag{A.6}$$

$$\frac{\partial\left(\overline{\rho}\widetilde{u_i}\right)}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_i u_j}\right) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \overline{\sigma_{ij}}}{\partial x_j} + \overline{\rho}\tilde{f}_i \tag{A.7}$$

$$\frac{\partial\left(\overline{\rho}\widetilde{E}\right)}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_j E}\right) = -\frac{\partial}{\partial x_j}\left(\overline{p u_j}\right) + \frac{\partial\left(\overline{u_i \sigma_{ij}}\right)}{\partial x_j} - \frac{\partial \overline{q_j}}{\partial x_j} + \overline{S_E} \tag{A.8}$$

As for the incompressible case to have equations in terms of the only filtered quantity it is necessary to apply the variable decompostion expressed by (2.44) and then apply again the Favre's trasformation:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_j}\right) = 0 \tag{A.9}$$

$$\frac{\partial\left(\overline{\rho}\widetilde{u_i}\right)}{\partial t}+\frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_i}\widetilde{u_j}\right)=-\frac{\partial\overline{p}}{\partial x_i}+\frac{\partial\widetilde{\sigma_{ij}}}{\partial x_j}+\overline{\rho}\tilde{f}_i+\underbrace{\frac{\partial}{\partial x_j}\left(\overline{\sigma_{ij}}-\widetilde{\sigma_{ij}}\right)}_{\text{I}}+\underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_i}\widetilde{u_j}-\rho\widetilde{u_iu_j}\right)}_{\text{II}} \tag{A.10}$$

$$\frac{\partial\left(\overline{\rho}\widetilde{E}\right)}{\partial t}+\frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_j}\widetilde{E}\right)=-\frac{\partial}{\partial x_j}\left(\overline{p}\widetilde{u_j}\right)+\frac{\partial\left(\widetilde{u_i}\widetilde{\sigma_{ij}}\right)}{\partial x_j}-\frac{\partial\widetilde{q_j}}{\partial x_j}+\overline{S_E}+\underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_j}\widetilde{E}-\overline{\rho}\widetilde{u_jE}\right)}_{\text{III}}+....$$

$$....+\underbrace{\frac{\partial}{\partial x_j}\left(\overline{p}\widetilde{u_j}-\overline{pu_j}\right)}_{\text{IV}}+\underbrace{\frac{\partial}{\partial x_j}\left(\overline{u_i\sigma_{ij}}-\widetilde{u_i}\widetilde{\sigma_{ij}}\right)}_{\text{V}}+\underbrace{\frac{\partial}{\partial x_j}\left(\widetilde{q_j}-\overline{q_j}\right)}_{\text{VI}} \tag{A.11}$$

Finally considering that $\widetilde{E}$ is defined as:

$$\widetilde{E}=c_v\widetilde{T}+\frac{1}{2}\overline{\rho}\widetilde{u_k}\widetilde{u_k}+\underbrace{\frac{1}{2}\overline{\rho}\left(\widetilde{u_ku_k}-\widetilde{u_k}\widetilde{u_k}\right)}_{\text{VII}} \tag{A.12}$$

The terms from I to VII needs to be modeled as the residuals stress tensor in the incompressible case to close the system of equations. This shows how the LES models become more complicated removing the incompressibility hypothesis.

Appendix B

OpenFOAM's near sonic case: blockMesh file, boundary and initial conditions

BlockMesh file

```
convertToMeters 0.0254;

vertices
(
  // block1 (nozzle)
  (0 0 0)   //0
  (7.74 0 0)//1
  (7.74 0.99996192306417100000 -0.00872653549837390000)   //2
  (0 2.99988576919251000000    -0.02617960649512170000)   //3
  (7.74 0.99996192306417100000  0.00872653549837390000)   //4
  (0 2.99988576919251000000     0.02617960649512170000)   //5

  // block2 (upper left)
  (-4.36 4.49982865378877000000  -0.03926940974268250000) //6
  (1.42  4.49982865378877000000  -0.03926940974268250000) //7
  (1.42  50.96805921858080000000 -0.44479151435211700000) //8
  (-4.36 50.19808853782140000000 -0.43807208201837000000) //9
  (-4.36 4.49982865378877000000   0.03926940974268250000) //10
  (1.42  4.49982865378877000000   0.03926940974268250000) //11
  (1.42  50.96805921858080000000  0.44479151435211700000) //12
  (-4.36 50.19808853782140000000  0.43807208201837000000) //13

  // block3 (upper middle left)
  (2.05 4.39983246148235000000  -0.03839675619284510000) //14
  (2.05 51.01805731473400000000 -0.44522784112703600000) //15
  (2.05 4.39983246148235000000   0.03839675619284510000) //16
  (2.05 51.01805731473400000000  0.44522784112703600000) //17

  // block4 (upper nozzle)
  (7.74 51.73802989934020000000 -0.45151094668586500000) //18
  (7.74 51.73802989934020000000  0.45151094668586500000)//19

```

```
33    // block5 (end up)
34    (80 0.99996192306417100000  -0.00872653549837390000) //20
35    (80 62.09763542228500000000 -0.54191785444901900000) //21
36    (80 0.99996192306417100000  0.00872653549837390000) //22
37    (80 62.09763542228500000000 0.54191785444901900000) //23
38
39    // block6 (end lower)
40    (80 0 0) //24
41
42
43  );
44
45  blocks
46  (
47      hex (0 1 2 3 0 1 4 5)
48      (97 97 1) simpleGrading (1 1 1) //block1
49      hex (6 7 8 9 10 11 12 13)
50      (11 168 1) simpleGrading (0.5 150 1)  //block2
51      hex (7 14 15 8 11 16 17 12)
52      (3 168 1) simpleGrading (0.5 150 1) //block3
53      hex (14 2 18 15 16 4 19 17)
54      (46 168 1) simpleGrading (0.435 150 1) //block4
55      hex (2 20 21 18 4 22 23 19)
56      (257 168 1) simpleGrading (8.6 150 1) //block5
57      hex (1 24 20 2 1 24 22 4)
58      (257 97 1) simpleGrading (8.6 1 1) //block6
59  );
60
61  edges
62  (
63      polyLine 5 4
64      ((0.83 2.98988614996187000000 0.02609234114013800000)
65      (1.9 2.79989338457968000000 0.02443429939544690000)
66      (4.22 1.90992727305257000000 0.01666768280189410000)
67      (6.07 1.21995354613829000000 0.01064637330801620000))
68      polyLine 3 2
69      ((0.83 2.98988614996187000000 -0.02609234114013800000)
70      (1.9 2.79989338457968000000 -0.02443429939544690000)
71      (4.22 1.90992727305257000000 -0.01666768280189410000)
72      (6.07 1.21995354613829000000 -0.01064637330801620000))
73  );
74
75
76  boundary
77  (
78      inlet
79      {
80          type patch;
81          faces
82          (
83              (0 3 5 0)
```

```
84          );
85      }
86      outlets
87      {
88          type patch;
89          inGroups (freestream);
90          faces
91          (
92               //block 2
93               (10 6 9 13)
94               (13 12 8 9)
95               //block 3
96               (12 17 15 8)
97               //block 4
98               (19 18 15 17)
99               //block 5
100              (19 23 21 18)
101              (22 20 21 23)
102              //block 6
103              (24 20 22 24)
104         );
105     }
106
107     outer_wall
108     {
109         type wall;
110         faces
111         (
112
113             (16 4 2 14)
114             (11 16 14 7)
115             (6 7 11 10)
116         );
117     }
118
119     nozzle_wall
120     {
121         type wall;
122         faces
123         (
124             (5 4 2 3)
125         );
126     }
127
128     wedgeFront
129     {
130          type wedge;
131          faces
132          (
133             //block 1
134             (0 1 4 5)
```

```
            //block 2
            (10 11 12 13)
            //block 3
            (11 16 17 12)
            //block 4
            (16 4 19 17)
            //block 5
            (4 22 23 19)
            //block 6
            (1 24 22 4)
        );
    }
    wedgeBack
    {
        type wedge;
        faces
        (
            //block1
            (0 1 2 3)
            //block2
            (6 7 8 9)
            //block3
            (7 14 15 8)
            //block4
            (14 2 18 15)
            //block5
            (2 20 21 18)
            //block6
            (1 24 20 2)
        );
    }
    symmetry_plane
    {
        type symmetryPlane;
        faces
        (
            //block 1
            (0 1 1 0)
            //block 6
            (1 24 24 1)
        );
    }


);

mergePatchPairs
(

);
```

U boundary and initial conditions

```
Uexternal          (3.54 0 0);

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform $Uexternal;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    freestream
    {
        type            waveTransmissive;
        field           U;
        gamma           1.4;
        fieldInf        $Uexternal;
    }

    nozzle_wall
    {
        type            noSlip;
    }

    outer_wall
    {
        type            noSlip;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
```

p boundary and initial conditions

```
pOut            1e5;

dimensions      [1 -1 -2 0 0 0 0];
```

```
4
5  internalField   uniform $pOut;
6
7  boundaryField
8  {
9      inlet
10     {
11         type           totalPressure;
12         p0             uniform 1.861e5;
13         value          uniform 1.861e5;
14     }
15
16     freestream
17     {
18         type             waveTransmissive;
19         field            p;
20         gamma            1.4;
21         fieldInf         $pOut;
22     }
23
24     nozzle_wall
25     {
26         type             zeroGradient;
27     }
28
29     outer_wall
30     {
31         type             zeroGradient;
32     }
33
34
35     symmetry_plane
36     {
37         type             symmetryPlane;
38     }
39
40     wedgeFront
41     {
42         type             wedge;
43     }
44
45     wedgeBack
46     {
47         type             wedge;
48     }
49     #includeEtc "caseDicts/setConstraintTypes"
50 }
```

T boundary and initial conditions

```
1  Texternal           294.4;
2
3  dimensions      [0 0 0 1 0 0 0];
```

```
4
5  internalField   uniform $Texternal;
6
7  boundaryField
8  {
9      inlet
10     {
11         type           totalTemperature;
12         gamma          1.4;
13         T0             uniform 294.4;
14     }
15
16     freestream
17     {
18         type             inletOutlet;
19         inletValue       uniform $Texternal;
20         value            uniform $Texternal;
21     }
22
23     nozzle_wall
24     {
25         type             zeroGradient;
26     }
27
28     outer_wall
29     {
30         type             zeroGradient;
31     }
32
33     symmetry_plane
34     {
35         type             symmetryPlane;
36     }
37
38     wedgeFront
39     {
40         type             wedge;
41     }
42
43     wedgeBack
44     {
45         type             wedge;
46     }
47
48     #includeEtc "caseDicts/setConstraintTypes"
49 }
```

ω boundary and initial condition

```
1  omegaInlet      61.48;
2
3  dimensions      [0 0 -1 0 0 0 0];
4
```

```
internalField   uniform 61.48;

boundaryField
{
    inlet
    {
        type            inletOutlet;
        inletValue      uniform $omegaInlet;
        value           uniform $omegaInlet;
    }

    freestream
    {
        type            inletOutlet;
        inletValue      uniform 61.48;
        value           uniform 61.48;
    }

    nozzle_wall
    {
        type            fixedValue;
        value           uniform 1e-8;
    }

    outer_wall
    {
        type            omegaWallFunction;
        value           uniform 61.48;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }


    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

ν_T boundary and initial condition

```
dimensions      [0 2 -1 0 0 0 0];

```

```
internalField   uniform 0.53;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0.53;
    }

    freestream
    {
        type            calculated;
        value           uniform 0.53;
    }

    nozzle_wall
    {
       type            fixedValue;
       value           uniform 0;
    }

    outer_wall
    {
        type            nutkWallFunction;
        value           uniform 0.53;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
       type             wedge;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

k boundary and initial conditions

```
kInlet          361.32;

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 361.32;
```

```
6
7  boundaryField
8  {
9      inlet
10     {
11         type            inletOutlet;
12         inletValue      uniform $kInlet;
13         value           uniform $kInlet;
14     }
15
16     freestream
17     {
18         type            inletOutlet;
19         inletValue      uniform 361.32;
20         value           uniform 361.32;
21     }
22
23     nozzle_wall
24     {
25        type            fixedValue;
26        value           uniform 1e-8;
27     }
28
29     outer_wall
30     {
31         type            kqRWallFunction;
32         value           uniform 361.32;
33     }
34
35     symmetry_plane
36     {
37         type            symmetryPlane;
38     }
39
40     wedgeFront
41     {
42         type            wedge;
43     }
44
45     wedgeBack
46     {
47        type             wedge;
48
49     }
50     #includeEtc "caseDicts/setConstraintTypes"
51 }
```

α_T boundary and initial conditions

```
1  dimensions      [1 -1 -1 0 0 0 0];
2
3  internalField   uniform 0.0;
4
```

```
boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0.0;
    }

    freestream
    {
        type            calculated;
        value           uniform 0.0;
    }

    nozzle_wall
    {
       type            fixedValue;
       value           uniform 0;
    }

    outer_wall
    {
        type            compressible::alphatWallFunction;
        value           uniform 0;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;

    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

Appendix C

Axisymmetric simulations: blockMesh files, boundary and initial conditions

C.1 Axisymmetric no-wall case

BlockMesh file

```
convertToMeters 1;

vertices
(
  // block1 (nozzle)
  (0 0 0)   //0
  (2.3607 0 0)//1
  (2.3607 0.30498838653457200000 -0.00266159332700404000) //2
  (0      0.91496515960371700000 -0.00798477998101212000) //3
  (2.3607 0.30498838653457200000 0.00266159332700404000)  //4
  (0      0.91496515960371700000 0.00798477998101212000)  //5

  // block5 (end up)
  (32.3607  0.30498838653457200000    -0.00266159332700404000 ) //6
  (32.3607  13.898328874072000000000  -0.12128887859651200000)//7
  (2.3607  9.999619230641710000000    -0.08726535498373900000) //8
  (32.3607  0.30498838653457200000     0.00266159332700404000 ) //9
  (32.3607  13.898328874072000000000   0.12128887859651200000)//10
  (2.3607  9.999619230641710000000     0.08726535498373900000) //11

  // block6 (end lower)
  (32.3607 0 0) //12


);

blocks
(
    hex (0 1 2 3 0 1 4 5) (150 120 1)
    simpleGrading (1 1 1) //block1
```

```
    hex (2 6 7 8 4 9 10 11) (619 178 1)
    simpleGrading (5 100 1) //block2
    hex (1 12 6 2 1 12 9 4) (619 120 1)
    simpleGrading (5 1 1) //block3
);

edges
(
 polyLine 5 4
((0.25315 0.91091531381530700000 0.007949437512243700000)
(0.4575 0.84836769552764300000  0.007403592716820410000)
(0.712 0.75447127095191700000 0.006584171033523110000)
(1.2871  0.58352778020409700000 0.005092369790076090000)
(1.85135 0.37208583157217800000 0.003247143858944930000
))

polyLine 3 2
((0.25315 0.91091531381530700000 -0.007949437512243700000)
(0.4575 0.84836769552764300000  -0.007403592716820410000)
(0.712 0.75447127095191700000 -0.006584171033523110000)
(1.2871  0.58352778020409700000 -0.005092369790076090000)
(1.85135 0.37208583157217800000 -0.003247143858944930000
))

);


boundary
(
    nozzle_inlet
    {
        type patch;
        faces
        (
            (0 3 5 0)
        );
    }


     coflow_inlet
    {
        type patch;
        faces
        (
            (2 8 11 4)
        );
    }

    outlets
    {
        type patch;
```

```
82          inGroups (freestream);
83          faces
84          (
85               (7 8 11 10)   //block2 up
86               (6 7 10 9)  //block3 end
87               (12 6 9 12) //block6
88          );
89      }
90
91      nozzle_wall
92      {
93          type wall;
94          faces
95          (
96
97              (2 3 5 4)
98
99          );
100     }
101
102
103     wedgeFront
104     {
105          type wedge;
106          faces
107          (
108             //block 1
109             (0 1 4 5)
110             //block 2
111             (4 9 10 11)
112             //block 3
113             (1 12 9 4)
114          );
115     }
116     wedgeBack
117     {
118         type wedge;
119         faces
120         (
121             //block1
122             (0 1 2 3)
123             //block2
124             (2 6 7 8)
125             //block 3
126             (1 12 6 2)
127         );
128     }
129     symmetry_plane
130     {
131          type symmetryPlane;
132          faces
```

```
            (
                //block 1
                (0 1 1 0)
                //block 6
                (1 12 12 1)
            );
    }
);
mergePatchPairs
(

);
```

U boundary and initial conditions

```
Uexternal          (252 0 0);

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform $Uexternal;

boundaryField
{
    nozzle_inlet
    {
        type            zeroGradient;
    }

    coflow_inlet
    {
        type            fixedValue;
        value           uniform (252 0 0);
    }

    outlets
    {
        type            waveTransmissive;
        field           U;
        gamma           1.4;
        fieldInf        252;
    }

    nozzle_wall
    {
        type            noSlip;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
```

```
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
```

p boundary and initial condition

```
pOut            23800;

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform $pOut;

boundaryField
{
    nozzle_inlet
    {
        type          totalPressure;
        p0            uniform 45430.3;
        value         uniform 45430.3;
    }


    coflow_inlet
    {
        type            zeroGradient;
    }

    outlets
    {
        type            waveTransmissive;
        field           p;
        gamma           1.4;
        fieldInf        $pOut;
    }

    nozzle_wall
    {
        type            zeroGradient;
    }


    symmetry_plane
    {
        type            symmetryPlane;
    }

```

```
    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
```

T boundary and initial condition

```
Texternal           219;

dimensions      [0 0 0 1 0 0 0];

internalField   uniform $Texternal;

boundaryField
{
    nozzle_inlet
    {
        type          totalTemperature;
        gamma         1.4;
        T0            uniform 696;
    }

    coflow_inlet
    {
        type            inletOutlet;
        inletValue      uniform $Texternal;
        value           uniform $Texternal;
    }


    outlets
    {
        type            inletOutlet;
        inletValue      uniform $Texternal;
        value           uniform $Texternal;
    }

    nozzle_wall
    {
        type            zeroGradient;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }
```

```
40
41      wedgeFront
42      {
43          type            wedge;
44      }
45
46      wedgeBack
47      {
48          type            wedge;
49      }
50
51      #includeEtc "caseDicts/setConstraintTypes"
52  }
```

ω boundary and initial condition

```
1   omegaInlet      7.92;
2
3   dimensions      [0 0 -1 0 0 0 0];
4
5   internalField   uniform ;
6
7   boundaryField
8   {
9       nozzle_inlet
10      {
11          type            inletOutlet;
12          inletValue      uniform $omegaInlet;
13          value           uniform $omegaInlet;
14      }
15
16
17      coflow_inlet
18      {
19          type            inletOutlet;
20          inletValue      uniform 4.16;
21          value           uniform 4.16;
22      }
23
24      outlets
25      {
26          type            inletOutlet;
27          inletValue      uniform 4.16;
28          value           uniform 4.16;
29      }
30
31      nozzle_wall
32      {
33          type            omegaWallFunction;
34          value           uniform 7.92;
35      }
36
37      symmetry_plane
```

```
    {
        type            symmetryPlane;
    }


    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

ν_T boundary and initial conditions

```
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 5.15;

boundaryField
{
    nozzle_inlet
    {
        type            calculated;
        value           uniform 9.82;
    }

    coflow_inlet
    {
        type            calculated;
        value           uniform 5.15;
    }

    outlets
    {
        type            calculated;
        value           uniform 5.15;
    }

    nozzle_wall
    {
        type            nutkWallFunction;
        value           uniform 9.82;
    }

    symmetry_plane
    {
        type            symmetryPlane;
```

```
34      }
35
36      wedgeFront
37      {
38          type            wedge;
39      }
40
41      wedgeBack
42      {
43         type             wedge;
44      }
45
46      #includeEtc "caseDicts/setConstraintTypes"
47  }
```

k boundary and initial condition

```
kInlet          894;

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 238.14;

boundaryField
{
    nozzle_inlet
    {
        type            inletOutlet;
        inletValue      uniform $kInlet;
        value           uniform $kInlet;
    }


    coflow_inlet
    {
        type            inletOutlet;
        inletValue      uniform 238.14;
        value           uniform 238.14;
    }

    outlets
    {
        type            inletOutlet;
        inletValue      uniform 238.14;
        value           uniform 238.14;
    }

    nozzle_wall
    {
        type            kqRWallFunction;
        value           uniform 864;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
       type             wedge;

```

```
    }
    #includeEtc "caseDicts/setConstraintTypes"
```

α_T boundary and initial condition

```
dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0.0;

boundaryField
{
    nozzle_inlet
    {
        type            calculated;
        value           uniform 0.0;
    }

    coflow_inlet
    {
        type            calculated;
        value           uniform 0.0;
    }

    outlets
    {
        type            calculated;
        value           uniform 0.0;
    }

    nozzle_wall
    {
        type            compressible::alphatWallFunction;
        value           uniform 0;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;

    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

C.2 Axisymmetric wall case

BlockMesh File

```
convertToMeters 1;

vertices
(
  // block1 (nozzle)
  (0 0 0)   //0
  (2.3607 0 0)//1
  (2.3607 0.30498838653457200000 -0.00266159332700404000)   //2
  (0      0.91496515960371700000 -0.00798477998101212000)   //3
  (2.3607 0.30498838653457200000  0.00266159332700404000)   //4
  (0      0.91496515960371700000  0.00798477998101212000)   //5

  // block2 (upper left)
  (-1.3298 1.54085158685891000000 -0.01344680808369860000) //6
  (0.4331  1.54085158685891000000 -0.01344680808369860000) //7
  (0.4331  9.74911414035251000000 -0.08507923017987300000) //8
  (-1.3298 9.52001296600573000000 -0.08307989452064110000) //9
  (-1.3298 1.54085158685891000000  0.01344680808369860000) //10
  (0.4331  1.54085158685891000000  0.01344680808369860000) //11
  (0.4331  9.74911414035251000000  0.08507923017987300000) //12
  (-1.3298 9.52001296600573000000  0.08307989452064110000) //13

  // block3 (upper middle left)
  (0.62525 1.50655307289096000000 -0.01314748948688070000) //14
  (0.62525 9.77408537561868000000 -0.08529715084861280000) //15
  (0.62525 1.50655307289096000000  0.01314748948688070000) //14
  (0.62525 9.77408537561868000000  0.08529715084861280000) //15

  // block4 (upper nozzle)
  (2.3607 9.99961923064171000000 -0.08726535498373900000) //18
  (2.3607 9.99961923064171000000  0.08726535498373900000) //19

  // block5 (end up)
  (32.3607 0.30498838653457200000  -0.00266159332700404000 ) //20
  (32.3607 13.898328874072000000000 -0.12128887859651200000)//21
  (32.3607 0.30498838653457200000   0.00266159332700404000 ) //22
  (32.3607 13.898328874072000000000  0.12128887859651200000)//23

  // block6 (end lower)
  (32.3607 0 0) //24


);

blocks
(
    hex (0 1 2 3 0 1 4 5) (120 110 1)
    simpleGrading (1 1 1) //block1
```

```
49      hex (6 7 8 9 10 11 12 13) (62 215 1)
50      simpleGrading (0.5 70 1)  //block2
51      hex (7 14 15 8 11 16 17 12) (10 215 1)
52      simpleGrading (1 70 1) //block3
53      hex (14 2 18 15 16 4 19 17) (88 215 1)
54      simpleGrading (1 70 1) //block4
55      hex (2 20 21 18 4 22 23 19) (838 215 1)
56      simpleGrading (3 70 1) //block5
57      hex (1 24 20 2 1 24 22 4) (838 110 1)
58      simpleGrading (3 1 1) //block6
59  );
60
61  edges
62  (
63      polyLine 5 4
64  ((0.25315 0.91091531381530700000 0.00794943751224370000)
65  (0.4575 0.84836769552764300000  0.00740359271682041000)
66  (0.712 0.75447127095191700000   0.00658417103352311000)
67  (1.2871  0.58352778020409700000 0.00509236979007609000)
68  (1.85135 0.37208583157217800000 0.00324714385894493000
69  ))
70
71      polyLine 3 2
72  ((0.25315 0.91091531381530700000 -0.00794943751224370000)
73  (0.4575 0.84836769552764300000  -0.00740359271682041000)
74  (0.712 0.75447127095191700000 -0.00658417103352311000)
75  (1.2871  0.58352778020409700000 -0.00509236979007609000)
76  (1.85135 0.37208583157217800000 -0.00324714385894493000
77  ))
78
79  );
80
81
82  boundary
83  (
84      ingresso
85      {
86          type patch;
87          faces
88          (
89              (0 3 5 0)
90          );
91      }
92      uscita
93      {
94          type patch;
95          inGroups (freestream);
96          faces
97          (
98               //block 2
99               (13 12 8 9)
```

```
            (10 6 9 13)
            //block 3
            (12 17 15 8)
            //block 4
            (19 18 15 17)
            //block 5
            (19 23 21 18)
            (22 20 21 23)
            //block 6
            (24 20 22 24)
        );
    }

    muro_noslip
    {
        type wall;
        faces
        (

            (16 4 2 14)
            (11 16 14 7)
            (6 7 11 10)
        );
    }

    muro_slip
    {
        type wall;
        faces
        (
            (5 4 2 3)
        );
    }

    wedgeFront
    {
         type wedge;
         faces
         (
            //block 1
            (0 1 4 5)
            //block 2
            (10 11 12 13)
            //block 3
            (11 16 17 12)
            //block 4
            (16 4 19 17)
            //block 5
            (4 22 23 19)
            //block 6
            (1 24 22 4)
```

```
        );
    }
    wedgeBack
    {
        type wedge;
        faces
        (
            //block1
            (0 1 2 3)
            //block2
            (6 7 8 9)
            //block3
            (7 14 15 8)
            //block4
            (14 2 18 15)
            //block5
            (2 20 21 18)
            //block6
            (1 24 20 2)
        );
    }
    symmetry_plane
    {
         type symmetryPlane;
         faces
         (
            //block 1
            (0 1 1 0)
            //block 6
            (1 24 24 1)
         );
    }
);

mergePatchPairs
(
);
```

U boundary and initial conditions

```
Uexternal         (252 0 0);

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform $Uexternal;

boundaryField
{
    nozzle_inlet
    {
        type            zeroGradient;
    }

```

```
    coflow_inlet
    {
        type            fixedValue;
        value           uniform (252 0 0);
    }

    outlets
    {
        type            waveTransmissive;
        field           U;
        gamma           1.4;
        fieldInf        252;
    }

    nozzle_wall
    {
        type            noSlip;
    }

    duct_wall
    {
        type            noSlip;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
```

p boundary and initial condition

```
pOut            23800;

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform $pOut;

boundaryField
{
    nozzle_inlet
    {
```

```
        type            totalPressure;
        p0              uniform 45430.3;
        value           uniform 45430.3;
    }


    coflow_inlet
    {
        type              zeroGradient;
    }

    outlets
    {
        type              waveTransmissive;
        field             p;
        gamma             1.4;
        fieldInf          $pOut;
    }

    nozzle_wall
    {
        type              zeroGradient;
    }

    duct_wall
    {
        type              zeroGradient;
    }


    symmetry_plane
    {
        type              symmetryPlane;
    }

    wedgeFront
    {
        type              wedge;
    }

    wedgeBack
    {
        type              wedge;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
```

T boundary and initial condition

```
Texternal           219;

dimensions      [0 0 0 1 0 0 0];

```

```
internalField   uniform $Texternal;

boundaryField
{
    nozzle_inlet
    {
        type           totalTemperature;
        gamma          1.4;
        T0             uniform 696;
    }

    coflow_inlet
    {
        type             inletOutlet;
        inletValue       uniform $Texternal;
        value            uniform $Texternal;
    }


    outlets
    {
        type             inletOutlet;
        inletValue       uniform $Texternal;
        value            uniform $Texternal;
    }

    nozzle_wall
    {
        type             zeroGradient;
    }

    duct_wall
    {
        type             zeroGradient;
    }

    symmetry_plane
    {
        type             symmetryPlane;
    }

    wedgeFront
    {
        type             wedge;
    }

    wedgeBack
    {
        type             wedge;
    }

```

```
    #includeEtc "caseDicts/setConstraintTypes"
}
```

ω boundary and initial condition

```
omegaInlet      7.92;

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform ;

boundaryField
{
    nozzle_inlet
    {
        type            inletOutlet;
        inletValue      uniform $omegaInlet;
        value           uniform $omegaInlet;
    }


    coflow_inlet
    {
        type            inletOutlet;
        inletValue      uniform 4.16;
        value           uniform 4.16;
    }

    outlets
    {
        type            inletOutlet;
        inletValue      uniform 4.16;
        value           uniform 4.16;
    }

    nozzle_wall
    {
        type            omegaWallFunction;
        value           uniform 7.92;
    }

    duct_wall
    {
        type            omegaWallFunction;
        value           uniform 4.16;
    }

    symmetry_plane
    {
        type            symmetryPlane;
    }


```

```
    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
        type            wedge;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

ν_T boundary and initial conditions

```
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 5.15;

boundaryField
{
    nozzle_inlet
    {
        type            calculated;
        value           uniform 9.82;
    }

    coflow_inlet
    {
        type            calculated;
        value           uniform 5.15;
    }

    outlets
    {
        type            calculated;
        value           uniform 5.15;
    }

    nozzle_wall
    {
        type            nutkWallFunction;
        value           uniform 9.82;
    }

    duct_wall
    {
        type            nutkWallFunction;
        value           uniform 5.15;
    }

    symmetry_plane
    {
```

```
        type            symmetryPlane;
    }

    wedgeFront
    {
        type            wedge;
    }

    wedgeBack
    {
       type             wedge;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}
```

k boundary and initial condition

```
kInlet          894;

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 238.14;

boundaryField
{
    nozzle_inlet
    {
        type            inletOutlet;
        inletValue      uniform $kInlet;
        value           uniform $kInlet;
    }


    coflow_inlet
    {
        type            inletOutlet;
        inletValue      uniform 238.14;
        value           uniform 238.14;
    }

    outlets
    {
        type            inletOutlet;
        inletValue      uniform 238.14;
        value           uniform 238.14;
    }

    nozzle_wall
    {
        type            kqRWallFunction;
        value           uniform 864;
    }
```

```
36
37      duct_wall
38      {
39          type            kqRWallFunction;
40          value           uniform 238.14;
41      }
42
43      symmetry_plane
44      {
45          type            symmetryPlane;
46      }
47
48      wedgeFront
49      {
50          type            wedge;
51      }
52
53      wedgeBack
54      {
55         type             wedge;
56
57      }
58      #includeEtc "caseDicts/setConstraintTypes"
```

α_T boundary and initial condition

```
1  dimensions      [1 -1 -1 0 0 0 0];
2
3  internalField   uniform 0.0;
4
5  boundaryField
6  {
7      nozzle_inlet
8      {
9          type            calculated;
10         value           uniform 0.0;
11     }
12
13     coflow_inlet
14     {
15         type            calculated;
16         value           uniform 0.0;
17     }
18
19     outlets
20     {
21         type            calculated;
22         value           uniform 0.0;
23     }
24
25     nozzle_wall
26     {
27         type            compressible::alphatWallFunction;
```

```
28          value           uniform 0;
29      }
30
31      duct_wall
32      {
33          type            compressible::alphatWallFunction;
34          value           uniform 0;
35      }
36
37      symmetry_plane
38      {
39          type            symmetryPlane;
40      }
41
42      wedgeFront
43      {
44          type            wedge;
45      }
46
47      wedgeBack
48      {
49          type            wedge;
50
51      }
52
53      #includeEtc "caseDicts/setConstraintTypes"
54  }
```

Appendix D

Gmsh files for the 3D mesh generation

D.1 No wall case .geo file

```
layer_rotation = 45;
//+
n_cell_nozzle_y = 100;
//+
n_cell_nozzle_x = 120;
//+
qy_nozzle = 1;
//+
qx_nozzle = 1;
//+
qx_12 = 1.0029;
//+
n_cell_two_x = 563;
//+
qy_3 = 1.0381;
//+
n_cell_three_y = 129;
//+
r_hole=0.138;
//+
n_cell_extruded_arc = 23;
//+
q_extruded_arc = 1;
//+
Point(1) = {0, r_hole, 0, 1.0};
//+
Point(2) = {2.3607, r_hole, 0, 1.0};
//+
Point(3) = {32.3607, r_hole, 0, 1.0};
//+
Point(4) = {2.3607, 0.305, 0, 1.0};
//+
Point(5) = {32.3607, 0.305, 0, 1.0};
```

```
//+
Point(6) = {32.3607, 13.89885811, 0, 1.0};
//+
Point(7) = {2.3607, 10, 0, 1.0};
//+
Point(8) = {0, 0.915, 0, 1.0};
//+
Point(9) = {0.25315, 0.89543, 0, 1.0};
//+
Point(10) = {0.4575, 0.8484, 0, 1.0};
//+
Point(11) = {1.2871, 0.58355, 0, 1.0};
//+
Point(12) = {1.85135, 0.3721, 0, 1.0};
//+
Line(1) = {1, 2};
//+
Line(2) = {2, 3};
//+
Line(3) = {3, 5};
//+
Line(4) = {4, 5};
//+
Line(5) = {7, 6};
//+
Line(6) = {5, 6};
//+
Line(7) = {4, 7};
//+
Line(8) = {1, 8};
//+
Line(9) = {2, 4};
//+
Spline(10) = {8, 9, 10, 11, 12,4};
//+
Curve Loop(1) = {1, 9, -10, -8};
//+
Surface(1) = {1};
//+
Curve Loop(2) = {2, 3, -4, -9};
//+
Surface(2) = {2};
//+
Curve Loop(3) = {4, 6, -5, -7};
//+
Surface(3) = {3};
//+
Transfinite Curve {9, 8, 3} = n_cell_nozzle_y+1
Using Progression qy_nozzle;
//number of points in the nozzle y
//+
```

```
Transfinite Curve {1, 10} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//number of points in the nozzle x
//+//+
Transfinite Curve {2, 4, 5} = n_cell_two_x+1
Using Progression qx_12;
//number of points x direction block 2
//+
Transfinite Curve {7, 6} = n_cell_three_y+1
Using Progression qy_3;
// number of points y direction all domain
//+
Transfinite Surface {1};
//+
Transfinite Surface {2};
//+
Transfinite Surface {3};
//+
Recombine Surface {2, 1, 3};
//+
Extrude {{1, 0, 0}, {0, 0, 0}, Pi/2} {
  Surface{1}; Surface{2}; Surface{3};
  Layers{layer_rotation}; Recombine;
}
//+
Extrude {{1, 0, 0}, {0, 0, 0}, Pi/2} {
  Surface{76}; Surface{32}; Surface{54};
  Layers{layer_rotation}; Recombine;
}
//+
Extrude {{1, 0, 0}, {0, 0, 0}, Pi/2} {
  Surface{120}; Surface{98}; Surface{142};
  Layers{layer_rotation}; Recombine;
}
//+
Extrude {{1, 0, 0}, {0, 0, 0}, Pi/2} {
  Surface{186}; Surface{164}; Surface{208};
  Layers{layer_rotation}; Recombine;
}
//+
Point(236) = {0, r_hole/2, 0 , 1.0};
//+
Point(237) = {0, -r_hole/2, 0, 1.0};
//+
Point(238) = {0, 0, r_hole/2, 1.0};
//+
Point(239) = {0, 0, -r_hole/2, 1.0};
//+
Point(240) = {2.3607, r_hole/2, 0 , 1.0};
//+
Point(241) = {2.3607, -r_hole/2, 0, 1.0};
```

```
//+
Point(242) = {2.3607, 0, r_hole/2, 1.0};
//+
Point(243) = {2.3607, 0, -r_hole/2, 1.0};
//+
Point(244) = {32.3607, r_hole/2, 0 , 1.0};
//+
Point(245) = {32.3607, -r_hole/2, 0, 1.0};
//+
Point(246) = {32.3607, 0, r_hole/2, 1.0};
//+
Point(247) = {32.3607, 0, -r_hole/2, 1.0};
//+
Line(259) = {236, 239};
//+
Line(260) = {237, 239};
//+
Line(261) = {238, 237};
//+
Line(262) = {236, 238};
//+
Line(263) = {236, 1};
//+
Line(264) = {238, 13};
//+
Line(265) = {101, 239};
//+
Line(266) = {237, 85};
//+
Line(267) = {243, 241};
//+
Line(268) = {242, 241};
//+
Line(269) = {240, 242};
//+
Line(270) = {243, 240};
//+
Line(271) = {240, 2};
//+
Line(272) = {243, 102};
//+
Line(273) = {241, 86};
//+
Line(274) = {242, 14};
//+
Line(275) = {247, 244};
//+
Line(276) = {246, 245};
//+
Line(277) = {247, 245};
//+
```

```
187 Line(278) = {244, 246};
188 //+
189 Line(279) = {126, 247};
190 //+
191 Line(280) = {245, 100};
192 //+
193 Line(281) = {246, 42};
194 //+
195 Line(282) = {244, 3};
196 //+
197 Line(283) = {236, 240};
198 //+
199 Line(284) = {237, 241};
200 //+
201 Line(285) = {238, 242};
202 //+
203 Line(286) = {239, 243};
204 //+
205 Line(287) = {240, 244};
206 //+
207 Line(288) = {241, 245};
208 //+
209 Line(289) = {242, 246};
210 //+
211 Line(290) = {243, 247};
212 // Cerchio 1 (inizio)
213 //+
214 Curve Loop(4) = {262, 264, -17, -263};
215 //+
216 Plane Surface(264) = {4};
217 //+
218 Curve Loop(5) = {261, 266, -105, -264};
219 //+
220 Plane Surface(265) = {5};
221 //+
222 Curve Loop(6) = {260, -265, -149, -266};
223 //+
224 Plane Surface(266) = {6};
225 //+
226 Curve Loop(7) = {259, -265, 236, -263};
227 //+
228 Plane Surface(267) = {7};
229 //+
230 Curve Loop(8) = {259, -260, -261, -262};
231 //+
232 Plane Surface(268) = {8};
233 //+
234 // Cerchio 2 (mezzo)
235 //+
236 Curve Loop(9) = {267, 273, 150, -272};
237 //+
```

```
Plane Surface(269) = {9};
//+
Curve Loop(10) = {268, 273, -106, -274};
//+
Plane Surface(270) = {10};
//+
Curve Loop(11) = {269, 274, -18, -271};
//+
Plane Surface(271) = {11};
//+
Curve Loop(12) = {237, -271, -270, 272};
//+
Plane Surface(272) = {12};
//+
Curve Loop(13) = {267, -268, -269, -270};
//+
Plane Surface(273) = {13};
//+
// Cerchio 3 (fine)
//+
Curve Loop(14) = {276, 280, -128, -281};
//+
Plane Surface(274) = {14};
//+
Curve Loop(15) = {194, 279, 277, 280};
//+
Plane Surface(275) = {15};
//+
Curve Loop(16) = {275, 282, -258, 279};
//+
Plane Surface(276) = {16};
//+
Curve Loop(17) = {282, 40, -281, -278};
//+
Plane Surface(277) = {17};
//+
Curve Loop(18) = {275, 278, 276, -277};
//+
Plane Surface(278) = {18};
//+
//quadrati_lunghi
//+
Curve Loop(19) = {289, -278, -287, 269};
//+
Plane Surface(279) = {19};
//+
Curve Loop(20) = {288, -276, -289, 268};
//+
Plane Surface(280) = {20};
//+
Curve Loop(21) = {277, -288, -267, 290};
```

```
//+
Plane Surface(281) = {21};
//+
Curve Loop(22) = {275, -287, -270, 290};
//+
Plane Surface(282) = {22};
//+
//quadrati_corti
//+
Curve Loop(23) = {261, 284, -268, -285};
//+
Plane Surface(283) = {23};
//+
Curve Loop(24) = {259, 286, 270, -283};
//+
Plane Surface(284) = {24};
//+
Curve Loop(25) = {262, 285, -269, -283};
//+
Plane Surface(285) = {25};
//+
Curve Loop(26) = {260, 286, 267, -284};
//+
Plane Surface(286) = {26};
//+
//ali_corte
//+
Curve Loop(27) = {12, -274, -285, 264};
//+
Plane Surface(287) = {27};
//+
Curve Loop(28) = {266, 100, -273, -284};
//+
Plane Surface(288) = {28};
//+
Curve Loop(29) = {265, 286, 272, -144};
//+
Plane Surface(289) = {29};
//+
Curve Loop(30) = {283, 271, -1, -263};
//+
Plane Surface(290) = {30};
//+
Curve Loop(31) = {272, 188, 279, -290};
//+
//ali_lunghe
//+
Plane Surface(291) = {31};
//+
Curve Loop(32) = {281, -34, -274, 289};
//+
```

```
340 Plane Surface(292) = {32};
341 //+
342 Curve Loop(33) = {271, 2, -282, -287};
343 //+
344 Plane Surface(293) = {33};
345 //+
346 Curve Loop(34) = {280, -122, -273, 288};
347 //+
348 Plane Surface(294) = {34};
349 //+
350 // parallelepipedo_corto
351 //+
352 Surface Loop(1) = {283, 284, 285, 286, 273, 268};
353 //+
354 Volume(13) = {1};
355 //+
356 // parallelepipedo_lungo
357 //+
358 Surface Loop(2) = {279, 280, 281, 282, 273, 278};
359 //+
360 Volume(14) = {2};
361 //+
362 //Volumi_ali_corte
363 //+
364 Surface Loop(3) = {264, 290, 287, 19, 285, 271};
365 //+
366 Volume(15) = {3};
367 //+
368 Surface Loop(4) = {283, 287, 288, 107, 265, 270};
369 //+
370 Volume(16) = {4};
371 //+
372 Surface Loop(5) = {266, 151, 289, 286, 288, 269};
373 //+
374 Volume(17) = {5};
375 //+
376 Surface Loop(6) = {267, 238, 290, 289, 284, 272};
377 //+
378 Volume(18) = {6};
379 //+
380 //Volume_ali_lunghe
381 //+
382 Surface Loop(7) = {270, 292, 294, 280, 129, 274};
383 //+
384 Volume(19) = {7};
385 //+
386 Surface Loop(8) = {275, 195, 294, 269, 281, 291};
387 //+
388 Volume(20) = {8};
389 //+
390 Surface Loop(9) = {271, 41, 292, 277, 293, 279};
```

```
//+
Volume(21) = {9};
//+
Surface Loop(10) = {291, 259, 293, 282, 272, 276};
//+
Volume(22) = {10};
//+
//
//Volume_13_tranfinite
//+
Transfinite Curve {283, 286, 284, 285} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {270, 267, 268, 269, 261, 262, 259, 260}
= layer_rotation+1 Using Progression 1;
//+
//Volume_14_trasfinite
//+
Transfinite Curve {288, 289, 287, 290} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {268, 269, 270, 267, 275, 276, 277, 278}
= layer_rotation+1  Using Progression 1;
//+
//Volume_15
//+
Transfinite Curve {12, 285, 283, 1} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {262, 17, 269, 18} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {271, 274, 264, 263} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_16
//+
Transfinite Curve {100, 284, 285, 12} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {105, 261, 268, 106} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {273, 274, 266, 264} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_17
//+
Transfinite Curve {100, 284, 144, 286} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
```

```
Transfinite Curve {149, 260, 150, 267} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {273, 272, 266, 265} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_18
//+
Transfinite Curve {283, 1, 286, 144} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {259, 236, 237, 270} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {272, 271, 265, 263} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_19
//+
Transfinite Curve {122, 288, 289, 34} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {106, 268, 128, 276} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {281, 280, 274, 273} =  n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume20
//+
Transfinite Curve {122, 288, 188, 290} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {150, 267, 194, 277} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {279, 280, 273, 272} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_22
Transfinite Curve {290, 188, 287, 2} =  n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {270, 237, 275, 258} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {279, 282, 272, 271} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
//Volume_13
//
```

```
Transfinite Surface {273};
//+
Transfinite Surface {284};
//+
Transfinite Surface {285};
//+
Transfinite Surface {286};
//+
Transfinite Surface {283};
//+
Transfinite Surface {268};
//Volume_14
//+
Transfinite Surface {273};
//+
Transfinite Surface {282};
//+
Transfinite Surface {280};
//+
Transfinite Surface {281};
//+
Transfinite Surface {279};
//+
Transfinite Surface {278};
//+
//Volume_15
//+
Transfinite Surface {264};
//+
Transfinite Surface {290};
//+
Transfinite Surface {19};
//+
Transfinite Surface {285};
//+
Transfinite Surface {287};
//+
Transfinite Surface {271};
//+
//Volume_16
//+
Transfinite Surface {287};
//+
Transfinite Surface {283};
//+
Transfinite Surface {265};
//+
Transfinite Surface {107};
//+
Transfinite Surface {288};
//+
```

```
Transfinite Surface {270};
//+
//Volume_17
//+
Transfinite Surface {266};
//+
Transfinite Surface {289};
//+
Transfinite Surface {286};
//+
Transfinite Surface {151};
//+
Transfinite Surface {288};
//+
Transfinite Surface {269};
//+
//Volume_18
//+
Transfinite Surface {267};
//+
Transfinite Surface {238};
//+
Transfinite Surface {284};
//+
Transfinite Surface {289};
//+
Transfinite Surface {290};
//+
Transfinite Surface {272};
//+
//Volume 19
//+
Transfinite Surface {274};
//+
Transfinite Surface {280};
//+
Transfinite Surface {294};
//+
Transfinite Surface {129};
//+
Transfinite Surface {270};
//+
Transfinite Surface {292};
//+
//Volume_20
//+
Transfinite Surface {271};
//+
Transfinite Surface {41};
//+
Transfinite Surface {292};
```

```
//+
Transfinite Surface {293};
//+
Transfinite Surface {279};
//+
Transfinite Surface {277};
//+
//Volume_21
//+
Transfinite Surface {275};
//+
Transfinite Surface {294};
//+
Transfinite Surface {195};
//+
Transfinite Surface {281};
//+
Transfinite Surface {291};
//+
Transfinite Surface {269};
//+
//Volume_22
//+
Transfinite Surface {272};
//+
Transfinite Surface {259};
//+
Transfinite Surface {282};
//+
Transfinite Surface {291};
//+
Transfinite Surface {293};
//+
Transfinite Surface {276};
//+
//Transfinite_Volume_13
//+
Transfinite Volume{13} = {238, 236, 239, 237, 242, 240, 243, 241};
//+
//Transfinite_Volume_14
//+
Transfinite Volume{14} = {242, 240, 243, 241, 246, 244, 247, 245};
//+
//Transfinite_Volume_15
//+
Transfinite Volume{15} = {13, 1, 236, 238, 14, 2, 240, 242};
//+
//Transfinite_volume_16
//+
Transfinite Volume{16} = {13, 238, 237, 85, 14, 242, 241, 86};
//+
```

```
//Transfinite_volume_17
//+
Transfinite Volume{17} = {85, 237, 239, 101, 86, 241, 243, 102};
//+
//Transfinite_volume_18
//+
Transfinite Volume{18} = {236, 1, 101, 239, 240, 2, 102, 243};
//+
//Transfinite_volume_19
//+
Transfinite Volume{19} = {14, 242, 241, 86, 42, 246, 245, 100};
//+
//Transfinite_volume_20
//+
Transfinite Volume{20} = {241, 243, 102, 86, 245, 247, 126, 100};
//+
//Transfinite_volume_21
//+
Transfinite Volume{21} = {14, 2, 240, 242, 42, 3, 244, 246};
//+
//Transfinite_volume_22
//+
Transfinite Volume{22} = {244, 3, 126, 247, 240, 2, 102, 243};
//+
Mesh.RecombineAll = 1 ;
//+
Physical Volume(1) = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22};
//+
Physical Surface(2) = {119, 163, 250, 31, 266, 267,
264, 265, 268};
//+
Physical Surface(3) = {75, 97, 185, 229};
//+
Physical Surface(4) = {177, 89, 67, 221, 133, 199, 45, 263,
275, 278, 274, 277, 276};
//+
Physical Surface(5) = {181, 71, 93, 225};
//+
Physical Surface(6) = {115, 159, 27, 246};
```

D.2 Wall case .geo file

```
layer_rotation = 45;
//+
qx_12 = 1.0036;
//+
qx_nozzle = 1;
//+
qy_nozzle = 1;
//+
```

```
n_cell_nozzle_y = 100;
//+
n_cell_nozzle_x = 110;
//+
n_cell_two_x = 502;
//+
n_cell_four_x = 81;
//+
n_cell_five_x = 64;
//+
n_cell_extruded_arc = 23;
//+
q_extruded_arc = 1;
//+
qx_4 = 1;
//+
qx_5 = 1.011;
//+
n_cell_three_y = 103;
//+
qy_3 = 1.0505;
//+
r_hole=0.138;
//+
Point(1) = {0, r_hole, 0, 1.0};
//+
Point(2) = {2.3607, r_hole, 0, 1.0};
//+
Point(3) = {32.3607, r_hole, 0, 1.0};
//+
Point(4) = {2.3607, 0.305, 0, 1.0};
//+
Point(5) = {32.3607, 0.305, 0, 1.0};
//+
Point(6) = {32.3607, 13.89885811, 0, 1.0};
//+
Point(7) = {2.3607, 10, 0, 1.0};
//+
Point(8) = {0.62525, 9.75519375, 0, 1.0};
//+
Point(9) = {-1.3298, 9.47941026, 0, 1.0};
//+
Point(10) = {-1.3298, 1.54091026, 0, 1.0};
//+
Point(11) = {0.62525, 1.54091026, 0, 1.0};
//+
Point(12) = {0, 0.915, 0, 1.0};
//+
Point(13) = {0.25315, 0.91095, 0, 1.0};
//+
Point(14) = {0.4575, 0.8484, 0, 1.0};
```

```
60  //+
61  Point(15) = {0.712, 0.7545, 0 , 1.0};
62  //+
63  Point(16) = {1.2871, 0.58355, 0, 1.0};
64  //+
65  Point(17) = {1.85135, 0.3721, 0, 1.0};
66  //+
67  Line(1) = {1, 2};
68  //+
69  Line(2) = {2, 3};
70  //+
71  Line(3) = {5, 6};
72  //+
73  Line(4) = {7, 6};
74  //+
75  Line(5) = {7, 8};
76  //+
77  Line(6) = {8, 9};
78  //+
79  Line(7) = {10, 9};
80  //+
81  Line(8) = {11, 8};
82  //+
83  Line(9) = {4, 7};
84  //+
85  Line(10) = {11, 10};
86  //+
87  Line(11) = {4, 11};
88  //+
89  Line(12) = {1, 12};
90  //+
91  BSpline(13) = {12, 13, 14, 15, 16, 17, 4};
92  //+
93  Line(14) = {2, 4};
94  //+
95  Line(15) = {3, 5};
96  //+
97  Line(16) = {4, 5};
98  //+
99  Curve Loop(1) = {13, -14, -1, 12};
100 //+
101 Plane Surface(1) = {1};
102 //+
103 Curve Loop(2) = {9, 4, -3, -16};
104 //+
105 Plane Surface(2) = {2};
106 //+
107 Curve Loop(3) = {2, 15, -16, -14};
108 //+
109 Plane Surface(3) = {3};
110 //+
```

```
Curve Loop(4) = {8, -5, -9, 11};
//+
Plane Surface(4) = {4};
//+
Curve Loop(5) = {7, -6, -8, 10};
//+
Plane Surface(5) = {5};
//+
Transfinite Curve {14, 12, 15} = n_cell_nozzle_y+1
Using Progression qy_nozzle;
//number of points in the nozzle y
//+
Transfinite Curve {1, 13} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//number of points in the nozzle x
//+
Transfinite Curve {2, 16, 4} =  n_cell_two_x+1
Using Progression qx_12;
//number of points x direction block 2
//+
Transfinite Curve {7, 8, 9, 3} = n_cell_three_y+1
Using Progression qy_3;
// number of points y direction all domain
//+
Transfinite Curve {5, 11} = n_cell_four_x+1
Using Progression qx_4;
//number of cell x direction block 4
//+
Transfinite Curve {6, 10} = n_cell_five_x +1
Using Progression qx_5;
//number of cell x direction block 5
//+
Transfinite Surface {1};
//+
Transfinite Surface {4};
//+
Transfinite Surface {5};
//+
Transfinite Surface {2};
//+
Transfinite Surface {3};
//+
Recombine Surface {5, 4, 2, 1, 3};
//+
Extrude {{1, 0, 0}, {-1.3298, 0, 0}, Pi/2} {
  Surface{5}; Surface{4}; Surface{2}; Surface{1};
  Surface{3}; Layers{layer_rotation}; Recombine;
}

//+
Extrude {{1, 0, 0}, {-1.3298, 0, 0}, Pi/2} {
```

```
  Surface{38}; Surface{60}; Surface{82}; Surface{104};
  Surface{126}; Layers{layer_rotation}; Recombine;
}
//+
Extrude {{1, 0, 0}, {-1.3298, 0, 0}, Pi/2} {
  Surface{236}; Surface{192}; Surface{148}; Surface{170};
  Surface{214}; Layers{layer_rotation}; Recombine;
}
//+
Extrude {{1, 0, 0}, {-1.3298, 0, 0}, Pi/2} {
  Surface{302}; Surface{324}; Surface{280}; Surface{258};
   Surface{346}; Layers{layer_rotation}; Recombine;
}
//+
Point(236) = {0, r_hole/2, 0 , 1.0};
//+
Point(237) = {0, -r_hole/2, 0, 1.0};
//+
Point(238) = {0, 0, r_hole/2, 1.0};
//+
Point(239) = {0, 0, -r_hole/2, 1.0};
//+
Point(240) = {2.3607, r_hole/2, 0 , 1.0};
//+
Point(241) = {2.3607, -r_hole/2, 0, 1.0};
//+
Point(242) = {2.3607, 0, r_hole/2, 1.0};
//+
Point(243) = {2.3607, 0, -r_hole/2, 1.0};
//+
Point(244) = {32.3607, r_hole/2, 0 , 1.0};
//+
Point(245) = {32.3607, -r_hole/2, 0, 1.0};
//+
Point(246) = {32.3607, 0, r_hole/2, 1.0};
//+
Point(247) = {32.3607, 0, -r_hole/2, 1.0};
//+
Line(447) = {245, 247};
//+
Line(448) = {244, 247};
//+
Line(449) = {245, 246};
//+
Line(450) = {246, 244};
//+
Line(451) = {241, 243};
//+
Line(452) = {240, 243};
//+
Line(453) = {240, 242};
```

```
//+
Line(454) = {242, 241};
//+
Line(455) = {239, 236};
//+
Line(456) = {238, 237};
//+
Line(457) = {239, 237};
//+
Line(458) = {238, 236};
//+
Line(459) = {239, 235};
//+
Line(460) = {238, 128};
//+
Line(461) = {236, 1};
//+
Line(462) = {237, 192};
//+
Line(463) = {242, 241};
//+
Line(464) = {241, 188};
//+
Line(465) = {242, 124};
//+
Line(466) = {240, 2};
//+
Line(467) = {243, 195};
//+
Line(468) = {246, 147};
//+
Line(469) = {244, 3};
//+
Line(470) = {247, 196};
//+
Line(471) = {245, 194};
//+
Line(472) = {236, 240};
//+
Line(473) = {237, 241};
//+
Line(474) = {238, 242};
//+
Line(478) = {239, 243};
//+
Line(479) ={240, 244};
//+
Line(480) = {241, 245};
//+
Line(481) = {242, 246};
//+
```

```
Line(482) = {243, 247};
//+
Curve Loop(6) = {449, 450, 448, -447};
//+
Plane Surface(452) = {6};
//+
Curve Loop(7) = {222, -471, 449, 468};
//+
Plane Surface(453) = {7};
//+
Curve Loop(8) = {112, -468, 450, 469};
//+
Plane Surface(454) = {8};
//+
Curve Loop(9) = {417, -469, 448, 470};
//+
Plane Surface(455) = {9};
//+
Curve Loop(10) = {471, 244, -470, -447};
//+
Plane Surface(456) = {10};
//+
Curve Loop(11) = {453, 465, -94, -466};
//+
Plane Surface(457) = {11};
//+
Curve Loop(12) = {416, -466, 452, 467};
//+
Plane Surface(458) = {12};
//+
Curve Loop(13) = {451, 467, -243, -464};
//+
Plane Surface(459) = {13};
//+
Curve Loop(14) = {204, -464, -454, 465};
//+
Plane Surface(460) = {14};
//+
Curve Loop(15) = {453, 454, 451, -452};
//+
Plane Surface(461) = {15};
//+
Curve Loop(16) = {460, -98, -461, -458};
//+
Plane Surface(462) = {16};
//+
Curve Loop(17) = {455, 461, -446, -459};
//+
Plane Surface(463) = {17};
//+
Curve Loop(18) = {457, 462, 340, -459};
```

```
315 //+
316 Plane Surface(464) = {18};
317 //+
318 Curve Loop(19) = {208, -462, -456, 460};
319 //+
320 Plane Surface(465) = {19};
321 //+
322 Curve Loop(20) = {458, -455, 457, -456};
323 //+
324 Plane Surface(466) = {20};
325 //+
326 // start big side surfaces
327 //+
328 Curve Loop(21) = {467, 238, -470, -482};
329 //+
330 Plane Surface(467) = {21};
331 //+
332 Curve Loop(22) = {469, -2, -466, 479};
333 //+
334 Plane Surface(468) = {22};
335 //+
336 Curve Loop(23) = {465, 106, -468, -481};
337 //+
338 Plane Surface(469) = {23};
339 //+
340 Curve Loop(24) = {-464, 480, 471, -216};
341 //+
342 Plane Surface(470) = {24};
343 //+
344 // end big side surfaces
345 //+
346 // start small side surfaces
347 //+
348 Curve Loop(25) = {-459, 478, 467, 328};
349 //+
350 Plane Surface(471) = {25};
351 //+
352 Curve Loop(26) = {466, -1, -461, 472};
353 //+
354 Plane Surface(472) = {26};
355 //+
356 Curve Loop(27) = {460, -86, -465, -474};
357 //+
358 Plane Surface(473) = {27};
359 //+
360 Curve Loop(28) = {-464, -473, 462, -196};
361 //+
362 Plane Surface(474) = {28};
363 //+
364 // end small side squares
365 //+
```

```
// start small squares
//+
Curve Loop(29) = {458, 472, 453, -474};
//+
Plane Surface(475) = {29};
//+
Curve Loop(30) = {451, -478, 457, 473};
//+
Plane Surface(476) = {30};
//+
Curve Loop(31) = {455, 472, 452, -478};
//+
Plane Surface(477) = {31};
//+
Curve Loop(32) = {456, 473, -454, -474};
//+
Plane Surface(478) = {32};
//+
// end small squares
// +
// start big squares
//+
Curve Loop(33) = {451, 482, -447, -480};
//+
Plane Surface(479) = {33};
//+
Curve Loop(34) = {448, -482, -452, 479};
//+
Plane Surface(480) = {34};
//+
Curve Loop(35) = {454, 480, 449, -481};
//+
Plane Surface(481) = {35};
//+
Curve Loop(36) = {450, -479, 453, 481};
//+
Plane Surface(482) = {36};
//+
// end big squares
//+
Surface Loop(1) = {461, 466, 476, 475, 478, 477};
//+
Volume(21) = {1};   // small square
//+
Surface Loop(2) = {461, 452, 482, 479, 481, 480};
//+
Volume(22) = {2}; // big square
//+
Surface Loop(3) = {341, 464, 459, 476, 471, 474};
//+
Volume(23) = {3}; // small side square NE
```

```
//+
Surface Loop(4) = {463, 447, 458, 477, 472, 471};
//+
Volume(24) = {4}; // small side square SE
//+
Surface Loop(5) = {462, 457, 473, 472, 475, 99};
//+
Volume(25) = {5}; // small side square SW
//+
Surface Loop(6) = {460, 465, 478, 209, 473, 474};
//+
Volume(26) = {6}; // small side square SE
//+
Surface Loop(7) = {459, 456, 245, 470, 467, 479};
//+
Volume(27) = {7};  // big side square NE
//+
Surface Loop(8) = {458, 418, 467, 468, 455, 480};
//+
Volume(28) = {8}; // big side square SE
//+
Surface Loop(9) = {454, 113, 482, 457, 468, 469};
//+
Volume(29) = {9}; // big side square NW
//+
Surface Loop(10) = {469, 470, 481, 223, 460, 453};
//+
Volume(30) = {10}; // big side square SW
//+
// Volume 21
// +
Transfinite Curve {456, 454, 453, 451, 452, 457, 455, 458}
= layer_rotation+1 Using Progression 1;
//+
Transfinite Curve {473, 474, 472, 478} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
// +
// end
//+
// Volume 22
//+
Transfinite Curve {451, 454, 453, 452, 447, 449, 450, 448}
= layer_rotation+1 Using Progression 1;
//+
Transfinite Curve {479, 481, 480, 482} = n_cell_two_x+1
Using Progression qx_12;
//+
// end
// +
// Volume 23
//+
```

```
Transfinite Curve {457, 340, 243, 451} = layer_rotation+1
Using Progression 1;
//+
Transfinite Curve {478, 328, 196, 473} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {467, 464, 462, 459} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
// end
//+
// Volume 24
//+
Transfinite Curve {328, 478, 472, 1} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {459, 461, 467, 466} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {416, 452, 455, 446} = layer_rotation+1
Using Progression 1;
//+
// end
//+
// Volume 25
//+
Transfinite Curve {474, 86, 472, 1} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {460, 461, 465, 466} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {94, 453, 458, 98} = layer_rotation+1
Using Progression 1;
//+
// end
// +
// Volume 26
//+
Transfinite Curve {473, 474, 86, 196} = n_cell_nozzle_x+1
Using Progression qx_nozzle;
//+
Transfinite Curve {464, 465, 462, 460} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {208, 456, 204, 454} = layer_rotation+1
Using Progression 1;
//+
// end
//+
// Volume 27
```

```
//+
Transfinite Curve {238, 482, 216, 480} = n_cell_two_x+1+1
Using Progression qx_12;
//+
Transfinite Curve {467, 464, 470, 471} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {447, 244, 243, 451} = layer_rotation+1
Using Progression 1;
//+
// end
//+
// Volume 28
//+
Transfinite Curve {238, 238, 482, 482, 479, 2} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {467, 466, 470, 469} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {417, 448, 416, 416, 452} = layer_rotation+1
Using Progression 1;
//+
// end
//+
// Volume 29
//+
Transfinite Curve {106, 481, 481, 479, 2} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {465, 466, 469, 468} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {112, 450, 453, 453, 94} = layer_rotation+1
Using Progression 1;
//+
// end
// +
// Volume 30
//+
Transfinite Curve {480, 481, 216, 106} = n_cell_two_x+1
Using Progression qx_12;
//+
Transfinite Curve {464, 465, 471, 468} = n_cell_extruded_arc+1
Using Progression q_extruded_arc;
//+
Transfinite Curve {449, 222, 204, 454} = layer_rotation+1
Using Progression 1;
//+
// end
// +
```

```
// Trasfinte Surfaces Volume 21
//+
Transfinite Surface {476};
//+
Transfinite Surface {461};
//+
Transfinite Surface {477};
//+
Transfinite Surface {478};
//+
Transfinite Surface {475};
//+
Transfinite Surface {466};
//+
// end
//+
// Transfinite Surfaces Volume 22
//+
Transfinite Surface {461};
//+
Transfinite Surface {479};
//+
Transfinite Surface {481};
//+
Transfinite Surface {452};
//+
Transfinite Surface {482};
//+
Transfinite Surface {480};
//+
// end
//+
// Transfinite Surfaces Volume 23
//+
Transfinite Surface {464};
//+
Transfinite Surface {471};
//+
Transfinite Surface {341};
//+
Transfinite Surface {476};
//+
Transfinite Surface {474};
//+
Transfinite Surface {459};
//+
// end
//+
// Transfinite Surfaces Volume 24
//+
Transfinite Surface {463};
```

```
//+
Transfinite Surface {472};
//+
Transfinite Surface {447};
//+
Transfinite Surface {477};
//+
Transfinite Surface {471};
//+
Transfinite Surface {458};
// +
// end
//+
// Transfinite Surfaces Volume 25
//+
Transfinite Surface {462};
//+
Transfinite Surface {472};
//+
Transfinite Surface {475};
//+
Transfinite Surface {99};
//+
Transfinite Surface {473};
//+
Transfinite Surface {457};
//+
// end
//+
// Transfinite Surfaces Volume 26
//+
Transfinite Surface {460};
//+
Transfinite Surface {209};
//+
Transfinite Surface {478};
//+
Transfinite Surface {465};
//+
Transfinite Surface {473};
//+
Transfinite Surface {474};
//+
// end
// +
// Transfinite Surfaces Volume 27
//+
Transfinite Surface {459};
//+
Transfinite Surface {467};
//+
```

```
Transfinite Surface {245};
//+
Transfinite Surface {479};
//+
Transfinite Surface {470};
//+
Transfinite Surface {456};
//+
// end
// +
// Transfinite Surfaces Volume 28
//+
Transfinite Surface {455};
//+
Transfinite Surface {418};
//+
Transfinite Surface {480};
//+
Transfinite Surface {458};
//+
Transfinite Surface {467};
//+
Transfinite Surface {468};
// +
// end
// +
// Transfinite Surfaces Volume 29
//+
Transfinite Surface {469};
//+
Transfinite Surface {482};
//+
Transfinite Surface {113};
//+
Transfinite Surface {468};
//+
Transfinite Surface {457};
//+
Transfinite Surface {454};
//+
// end
//+
// Transfinite surfaces Volume 30
//+
Transfinite Surface {453};
//+
Transfinite Surface {223};
//+
Transfinite Surface {470};
//+
Transfinite Surface {481};
```

```
//+
Transfinite Surface {469};
//+
Transfinite Surface {460};
// +
// end
// +
// Transfinite Volumes
//+
Transfinite Volume{21} = {242, 241, 243, 240, 238, 237, 239, 236};
//+
Transfinite Volume{22} = {246, 245, 247, 244, 242, 241, 243, 240};
//+
Transfinite Volume{23} = {235, 192, 237, 239, 195, 188, 241, 243};
//+
Transfinite Volume{24} = {1, 236, 239, 235, 2, 240, 243, 195};
//+
Transfinite Volume{25} = {124, 242, 240, 2, 128, 238, 236, 1};
//+
Transfinite Volume{26} = {124, 188, 241, 242, 128, 192, 237, 238};
//+
Transfinite Volume{27} = {245, 194, 196, 247, 241, 188, 195, 243};
//+
Transfinite Volume{28} = {3, 244, 247, 196, 2, 240, 243, 195};
//+
Transfinite Volume{29} = {147, 246, 244, 3, 124, 242, 240, 2};
//+
Transfinite Volume{30} = {147, 246, 245, 194, 124, 242, 241, 188};
//+
// end
//+
Mesh.RecombineAll = 1 ;
//+
Physical Volume(1) = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
 11, 12, 13, 14, 15, 16, 17, 18, 19,
 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30};
//+
Physical Surface(2) = {466, 464, 463, 462, 465, 451, 103, 345, 213};
//+
Physical Surface(3) = {355, 25, 135, 289};
//+
Physical Surface(4) = {73, 183, 401, 271,315, 380, 51,
161, 139, 29, 359, 293};
//+
Physical Surface(5) = {275, 405, 187, 77, 452, 456,
453, 454, 455, 117, 227, 249, 422};
//+
Physical Surface(6) = {147, 37, 301, 367, 59, 323, 169, 388};
//+
Physical Surface(7) = {333, 439, 201, 91};
```

Bibliography

[1] NASA, "Axysimmetric near-sonic jet validation case." "https://turbmodels.larc.nasa.gov/jetnearsonic_val.html, (Accessed May 13, 2020).

[2] G. A. Brès, P. Jordan, V. Jaunet, M. Le Rallic, A. V. G. Cavalieri, A. Towne, S. K. Lele, T. Colonius, and O. T. Schmidt, "Importance of the nozzle-exit boundary-layer state in subsonic turbulent jets," *Journal of Fluid Mechanics*, p. 83–124, 2018.

[3] M. Zhu, A. C. Pérez, P. Fosso, M. Sanjosé, and S. Moreau, "Isothermal and heated subsonic jet noise using large eddy simulations on unstructured grids," *Computers and Fluids*, vol. 171, pp. 166 – 192, 2018.

[4] R. Sandberg, N. Sandham, and V. Suponitsky, "Dns of fully turbulent jet flows in flight conditions including a canonical nozzle," *17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference)*.

[5] H. K. Versteeg and W. Malalasekera, "Conservation laws of fluid motion and boundary conditions," in *An Introduction to Computational Fluid Dynamics*, ch. 2, pp. 9–39, Edinburgh Gate, Harlow Essex CM20 2JE, England: PEARSON - Prentice Hall, 2007.

[6] S. Pope, "Direct numerical simulation," in *Turbulent Flows*, ch. 9, pp. 344–357, Cambridge University Press, 2001.

[7] S. Pope, "The scales of turbulent motion," in *Turbulent Flows*, ch. 6, pp. 182–263, Cambridge University Press, 2001.

[8] S. Pope, "Turbulent viscosity models," in *Turbulent Flows*, ch. 10, pp. 358–386, Cambridge University Press, 2001.

[9] S. Pope, "Reynolds-stress and related models," in *Turbulent Flows*, ch. 11, pp. 387–462, Cambridge University Press, 2001.

[10] D. C. Wilcox, "Effects of compressibility," in *Turbulence modeling for CFD*, ch. 5, pp. 239 – 297, DCW Industies, 2006.

[11] R. Paoli, *Equations notes ME - 518 "Fundamentals of turbulence"*. Mechanical Engineering Department, University of Illinois at Chicago, 2019.

[12] H. K. Versteeg and W. Malalasekera, "Turbulnce and its modeling," in *An Introduction to Computational Fluid Dynamics*, ch. 3, pp. 40–113, Edinburgh Gate, Harlow Essex CM20 2JE, England: PEARSON - Prentice Hall, 2007.

[13] D. C. Wilcox, "One-equation and two-equation models," in *Turbulence modeling for CFD*, ch. 4, pp. 107 – 229, DCW Industies, 2006.

[14] J. . Hinze, *Turbulence: An introduction to its Mechanism and Theory*. Technological University Delft, Holland: McGrawn-Hill, 1957.

[15] S. Pope, "Large-eddy simulation," in *Turbulent Flows*, ch. 13, pp. 558–640, Cambridge University Press, 2001.

[16] P. Sagaut, S. Deck, and M. Terracol, *Multiscale and Multiresolution approaches in turbulence*. 57 Shelton Street Covent Garden, London, UK: Imperial College Press, 2006.

[17] S. Ghosal, T. S. Lund, P. Moin, and K. Akselvoll, "A dynamic localization model for large-eddy simulation of turbulent flow," *Journal of Computational Physics*, vol. 125, pp. 187–206, 1996.

[18] C. Fuerby, "On subgrid scale modeling in large eddy simulations of compressible fluid flow," *Physics of Fluids*, vol. 8, no. 2, pp. 1301–1311, 1996.

[19] M. Germano, U. Piomelli, P. Moin, and W. Cabot, "A dynamic subgrid-scale eddy viscosity model," *Physics of Fluids*, vol. 3, no. 7, pp. 1760–1765, 1991.

[20] D. K. Lilly, "A proposed modification of the germano subgrid-scale closure method," *Physics of Fluids*, vol. 4, no. 3, pp. 633–635, 1992.

[21] R. Pomraning, *PhD dissertation "Development of Large Eddy Simulation Turbulence Models"*. Mechanical Engineering Department, University of Wisconsin-Madison, 2000.

[22] A. Travin, M. Shur, M. Strelets, and P. Spalart, "Detached-eddy simulation past a circular cilynder," *International Journal of Turbulence and Combustion*, vol. 23, pp. 293–313, 2000.

[23] U. Piomelli, E. Balaras, K. Squires, and P. Spalart, "Interaction of the inner and outer layers in large eddy simulations with wall-layer models," *International Journal of Heat and Fluid Flows*, vol. 24, pp. 538–550, 2003.

[24] C. J. Greenshields, *OpenFOAM-7 User Guide*. OpenFOAM Foundation Ltd., 2019.

[25] J. H. Ferziger and M. Peric, "Introduction to numerical methods," in *Computational Methods for Fluid Dynamics*, ch. 2, pp. 21–63, Springer, 2002.

[26] Y. Saad, *Iterative methods for sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[27] S. Yoon and A. Jameson, *Lower-Upper Symmetric-Gauss-Seidel method for the Euler and Navier- Stokes equations*. AIAA Journal 26 (9).

[28] J. H. Ferziger and M. Peric, "Solution of linear equation systems," in *Computational Methods for Fluid Dynamics*, ch. 5, pp. 91–129, Springer, 2002.

[29] J. H. Ferziger and M. Peric, "Solution of the navier-stokes equations," in *Computational Methods for Fluid Dynamics*, ch. 7, pp. 157–206, Springer, 2002.

[30] F. Moukalled, L. Mangani, and M. Darwish, "Fluid flow computation:compressible flow," in *The Finite Volume Method in Computational Fluid Dynamics An Advanced Introduction with OpenFOAM and Matlab*, ch. 7, pp. 655–689, Springer, 2016.

[31] J. Bridges and M. Wernet, "Establishing consensus turbulence statistics for hot subsonic jets," *AIAA Paper 16th AIAA/CEAS Aeroacoustics Conference, Stockholm Sweden*, June 2010.

[32] J. Bridges and M. Wernet, "The nasa subsonic jet particle image velocimetry (piv) dataset," *NASA/TM 2011 216807*, November 2011.

[33] F. Menter, "Improved two-equation k-omega turbulence model for aerodynamic flows," *NASA TM 103975*, October 1992.

[34] S. Pope, "Free shear flows," in *Turbulent Flows*, ch. 5, pp. 96–158, Cambridge University Press, 2001.

[35] R. Miake-lye, M. Martinez-Sanchez, R. C. Brown, and C. . Kolb, "Plume and wake dynamics, mixing and chemistry," *Journal of Aircraft*, vol. 30, no. 4, pp. 467 – 470, 1993.

[36] F. Garnier, C. Baudoin, P. Woods, and N. Louisnard, "Engine emission alteration in the near field of an aircraft," *Atmospheric Environment*, vol. 31, no. 12, pp. 1767 – 1781, 1997.

[37] "UIC Mechanical and Industrial Engineering Dragon documentation." `"http://dragon.mie.uic.edu/index.php/Main_Page`, (Accessed May 26, 2020).

[38] J. Lee and V. Chu, "Turbulent round jet in coflow," in *Turbulent jets and plumes - A lagrangian approach*, ch. 5, pp. 179–203, Kluwer Academic Publisher, 2003.

[39] B. Zang, U. Vevek, and T. New, "OpenFOAM-based numerical simulation study of an underexpanded supersonic jet." 55th AIAA Aerospace Sciences Metting, Grapevine, Texas `"http://dx.doi.org/10.2514/6.2017-0747`, January 9-13,2017.

[40] C. Geuzaine and J. Remacle, "Gmsh: a three dimensional finite element mesh generator with built-in pre and post- processing facilities," *International Journal for numerical methods in Engineering*, vol. 71, no. 11, pp. 1309 – 1311, 2009.

[41] "Gmsh official website." `"https://gmsh.info/`, (Accessed June 3, 2020).

[42] "Theta| Argonne Leadership Computing Facility." `"https://www.alcf.anl.gov/support-center/theta`, (Accessed June 9, 2020).

[43] D. Lindblab, A. Jareteg, and O. Petit, "Implementation and run-time mesh refinement for the $k-\omega$ SST DES turbulence model when applied to airfoils," *CFD with OpenSorce Software - A course at Chalmers University of Technology*, 2014.

[44] V. Kolàr̃, "Vortex identification:new requirements and limitations," *International Journal of Heat and Fluid flow*, pp. 638 – 652, 2007.

[45] J. Hunt, A. Wray, and P. Moin, "Eddies stream nd convergence zones in turbulent flows," *Center for Turbulence Research Report CTR-S88*, pp. 193 – 208.

[46] I. Sofia Larsson, T. Staffan Lundström, and B. Marjavaara, "Calculation of Klin Aerodynamics with two Rans Turbulence Models and by Ddes," *Flow Turbulence Combust*, vol. 94, pp. 859 – 878, 2015.