SAPIENZA
UNIVERSITÀ DI ROMA

Development of a high-fidelity OpenFOAM solver for aircraft aerodynamics

Department of Mechanical and Aerospace Engineering
PhD in Aeronautical and Space Engineering (XXXVII cycle)

Daniele Fontana
ID number 1635256

Advisor
Prof. Sergio Pirozzoli

Co-Advisor
Dott. Ivan Spisso

Academic Year 2024/2025

Thesis not yet defended

Development of a high-fidelity OpenFOAM solver for aircraft aerodynamics

PhD thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: daniele.fontana@uniroma1.it

Dedicated to
Ilaria

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020, risorse
FSE REACT-EU
Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione”
e Azione IV.5 “Dottorati su tematiche Green”

Abstract

The development of the `rhoEnergyFoam` solver represents a significant advancement in the numerical simulation of compressible turbulent flows, aimed at bridging the gap between high-accuracy academic methods and general-purpose Computational Fluid Dynamics (CFD) solvers used in industry. Leveraging an energy-consistent, low-diffusive numerical flux and selectively incorporating diffusion based on flow characteristics, `rhoEnergyFoam` ensures minimal numerical dissipation while maintaining robustness. The solver utilizes a novel triple-splitting approach, previously unexplored in finite-volume methods, which through the AUSM (Advection Upstream Splitting Method) decomposes advection fluxes of the Navier-Stokes equations into distinct pressure and convective components. This formulation allows for tailored numerical diffusion in different flow regimes: fully resolved simulations (Mode A) operating without artificial diffusion, unresolved smooth flows (Mode B) using minimal pressure diffusion, and shocked flows (Mode C) applying both pressure and convective diffusion for stability. Implemented within the OpenFOAM framework, `rhoEnergyFoam` offers a robust, highly-accurate tool for simulating compressible flows across a wide range of Mach and Reynolds numbers, advancing both fundamental turbulence modeling and real-world engineering applications. The solver's low-dissipation characteristics improve the accuracy of turbulence simulations, preserving kinetic energy in unsteady RANS and Large Eddy Simulations (LES), and reducing the numerical damping that can obscure physical flow features. Additionally, its shock-capturing capability, based on a local shock sensor, provides sharper resolution of supersonic flows—crucial for aerospace and automotive applications. Validation through a variety of fundamental and applied test cases demonstrates that `rhoEnergyFoam` outperforms standard OpenFOAM solvers in preserving kinetic energy and provides enhanced predictions of aerodynamic coefficients such as lift and drag. By maintaining minimal numerical diffusion while enabling selective diffusion where necessary, it ensures high-fidelity flow predictions without sacrificing stability. Furthermore, as an open-source solver, `rhoEnergyFoam` fosters collaboration and further development within the CFD community, encouraging innovation and improvement. As a potential next step, the solver could benefit from GPU acceleration to further enhance computational efficiency and speed, broadening its applicability to large-scale industrial simulations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The simulation of turbulent compressible flows remains a cornerstone in Computational Fluid Dynamics (CFD), requiring high-fidelity solvers capable of capturing complex flow behaviors in industrial and academic settings. One of the main limiting factors in the development of fixed-wing and rotary-wing air transport is the acoustic impact on the communities surrounding the airport areas. In this regard, European regulations provide for a reduction in pollution of about 50% of the noise level generated over the next 10 years. In order to substantially reduce the noise and environmental impact of aviation, in parallel with advances in propulsion technologies, the full understanding of the mechanisms related to the generation of noise (for example associated with the transition of flow from laminar to turbulent, at completely turbulent flow, to high-lift devices, to landing gears, etc.) remains an open problem, especially for future aircraft configurations that rely on innovative technologies. Therefore, research activities in aircraft aeroacoustics are currently focused on the following areas:

- Compatibility between thrust systems (electrically or mechanically driven), and higher degree of integration in unconventional configurations (e.g. boundary layer ingestion systems, distributed wing propulsion);

- Better understanding of the mechanisms of noise sources in high bypass ratio turbofan engines for advanced and disruptive aeronautical architectures, such as distributed propulsion, and tighter engine/wing integration;

- Better understanding of the noise generation mechanism associated with the transition of flow from laminar to turbulent state, and to fully turbulent flow regimes;

- Improved understanding of the entry/exit edge noise generated on landing by high-lift devices and landing gear, including new acoustic treatments on nacelles and primary aircraft structures.

This work introduces an advanced numerical solver for the compressible Navier-Stokes equations, prioritizing low numerical diffusion and enhanced energy conservation. The core of this solver lies in its use of an energy-preserving flux computation combined with the Advection Upstream Splitting Method (AUSM) for shock regions. This novel integration seeks to address the inherent limitations of many industrial CFD solvers, which rely on first or second-order discretization methods stabilized through upwind schemes and implicit segregated algorithms. These traditional methods often exhibit excessive numerical diffusion, which compromises their ability to resolve fine-scale turbulent structures, especially in scenarios such as large-eddy simulations (LES) or direct numerical simulations (DNS). By contrast, the solver described in this study incorporates features common in academic approaches, such as high-order discretization and explicit time integration, to achieve a balance between computational efficiency and predictive accuracy.
The mathematical foundation of the solver begins with the discretization of the compressible Navier-Stokes equations. The governing equations, integrated over an arbitrary control volume, are expressed in terms of conservative variables, convective fluxes, and viscous fluxes. The numerical fluxes are split into convective and pressure components, with the central part evaluated to preserve kinetic energy. This formulation ensures discrete conservation of total kinetic energy in the inviscid limit, a property essential for accurate turbulence modeling. The central flux, however, is insufficient for practical applications involving shocks or unstructured meshes. To address this, the solver employs a localized addition of numerical diffusion derived from the AUSM scheme. This diffusion is activated dynamically based on a shock sensor that evaluates local smoothness, ensuring minimal dissipation in smooth regions while effectively capturing shocks. The shock sensor uses velocity and length scales to detect zones requiring additional numerical stabilization, maintaining stability without compromising accuracy.

Key to the solver's performance is its innovative handling of numerical diffusion. By separating the AUSM flux into pressure and convective components, the solver minimizes diffusion in shock-free regions, ensuring that turbulence features are not overly damped. This is particularly advantageous in LES and DES, where maintaining turbulence fidelity is essential for accurate predictions. The use of adaptive diffusion, guided by the shock sensor, ensures stability without unnecessary dissipation, functioning as an embedded filter that activates only when numerical oscillations are detected. This localized approach contrasts with traditional methods that apply uniform diffusion across the computational domain, often leading to excessive damping of critical flow features.

The solver's design also emphasizes computational efficiency. While high-order methods and explicit time-stepping are traditionally associated with increased computational costs, the solver mitigates these through its modular operational modes and adaptive diffusion strategy. This makes it a viable tool for industrial applications, where balancing accuracy and efficiency is paramount.

The integration of the solver within OpenFOAM, a widely adopted open-source CFD framework, highlights its practical applicability. OpenFOAM's versatility facilitates the seamless incorporation of the proposed numerical methods, utilizing its built-in primitives for spatial discretization and boundary condition management. The solver is designed with three distinct operational modes to accommodate various flow regimes: Mode A for fully resolved smooth flows, Mode B for unresolved smooth flows, and Mode C for shocked flows. This structured approach allows users to tailor the solver's functionality to the specific requirements of their simulations, ensuring an optimal balance between computational efficiency and accuracy. Furthermore, its OpenFOAM implementation enhances accessibility, enabling researchers and engineers to integrate the solver into existing workflows with minimal adjustments.

The solver's capabilities are validated through a comprehensive set of test cases and its application extends beyond canonical flows to practical engineering scenarios. The solver's shock-capturing capabilities are tested using challenging flow configurations, including transonic and supersonic flows test cases

The PhD project will be conducted in close synergy with the partner Company, thus favoring the birth and development of specific internal skills. In terms of industrial repercussions, the improvement of the predictive capacity of aeroacoustic analysis codes is an aspect of fun-

damental importance in the design phase of new generation transport aircraft, and could lead to substantial competitive advantages in the medium term.

Chapter 2

Literature Review

The purpose of this chapter is to provide the theoretical and computational foundation necessary to understand the structure and capability of rhoEnergyFoam, the density-based solver described in this thesis. This new OpenFOAM solver is a powerful tool for CFD simulations due to its energy-preserving and shock-capturing properites. By exploring key concepts in fluid dynamics, numerical methods, and the computational framework used, this chapter aims to equip the reader with the background knowledge essential for comprehending the following chapters.
To achieve this, the present chapter begins with an overview of the Navier-Stokes equations, which govern fluid motion and it is at the basis of all fluid dynamics. Then, it discusses the numerical framework provided by OpenFOAM and the principles of the finite volume method (FVM) employed for discretizing these equations. The chapter also explores critical components of solver design, including interpolation schemes, flux computation using the Advection Upstream Splitting Method (AUSM), and strategies for energy preservation and shock capturing.

2.1 Navier-Stokes equations

The Navier-Stokes equations are a set of partial differential equations that describe the motion of fluid substances such as liquids and gases. These equations are fundamental to fluid mechanics and form the basis for computational fluid dynamics (CFD). The system comprises

three main equations: the **continuity equation**, the **momentum equation**, and the **energy equation**.

Continuity Equation
This equation represents conservation of mass in a flowing fluid expressed in conservative form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity vector, and t is time. The continuity equation states that the rate at which mass enters a system is equal to the rate at which mass leaves the system plus the accumulation of mass within the system. The time derivative can be seen as the accumulation (or loss) of mass in the system, while the divergence term represents the difference between the inflow and the outflow.

Momentum Equation
The momentum equation is the generalized model of Newton's Second Law of Motion applied to a fluid element. It describes how the velocity of the fluid changes over time and space due to the forces acting on it, such as pressure gradients, viscous stresses, and external forces like gravity:

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g} \tag{2.2}$$

where $\mathbf{g}$ represents body forces (e.g., gravity) and the stress tensor is defined as

$$\boldsymbol{\tau} = \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) - \frac{2}{3} \mu (\nabla \cdot \mathbf{u}) \mathbf{I}, \tag{2.3}$$

where μ is the dynamic viscosity, $\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right)$ is the rate of strain tensor, $\left(\frac{2}{3} \mu (\nabla \cdot \mathbf{u}) \mathbf{I} \right)$ is the bulk viscosity correction and $\mathbf{I}$ the identity matrix.
The left-hand side describes the total acceleration of the fluid, including local changes in velocity and the effects of fluid motion. The right-hand side represents forces acting on the fluid. Pressure forces (∇p) push the fluid from regions of high pressure to low pressure.

Viscous forces ($\nabla \cdot \tau$) arise from fluid friction, resisting motion. Body forces like gravity ($\rho \mathbf{g}$) pull the fluid downward or in the direction of external acceleration.

Energy Equation
The energy equation is a mathematical representation of the conservation of energy principle. It describes how the total energy of a fluid system evolves due to internal energy changes, work done by forces, heat transfer, and viscous dissipation.

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot [\mathbf{u}(\rho E + p)] = \nabla \cdot (k \nabla T) + \nabla \cdot (\mathbf{u} \cdot \boldsymbol{\tau}) - \nabla \cdot \mathbf{q} \quad (2.4)$$

where $E = e + \frac{1}{2}|\mathbf{u}|^2$ is the total specific energy, e is the internal energy, $|\mathbf{u}|$ is the velocity magnitude, k is the thermal conductivity, T is the temperature and $\mathbf{q}$ is the heat flux vector.
These equations are solved numerically in CFD to analyze fluid flows where changes in pressure, temperature, and density significantly affect the dynamics, such as supersonic aircraft, rocket engines, weather modeling and shock waves.

2.2 OpenFOAM

OpenFOAM (Open source Field Operation And Manipulation) is the leading open source software for Computational Fluid Dynamics (CFD), which is developed and constantly updated by the ESI-OpenCFD [28] ltd since 2004. It is distributed exclusively from the OpenFOAM Foundation [13] under the General Public Licence (GPL). The GPL gives users the freedom to modify and redistribute the software and a guarantee of continued free use, within the terms of the licence. The structure of OpenFOAM can be summarized in Figure 2.1. In fact, OpenFOAM is mainly an object-oriented C++ framework: it consists in a set of libraries containing a list of executables called "solvers". Each solver has a unique name which is referred to a specific physical sector (e.g. incompressible, compressible or multiphase flows, heat transfer analysis or chemical reactions) and it allows the resolution of fluid-dynamic equations. The software open source nature helps the creation of personalized solvers in a small amount of time

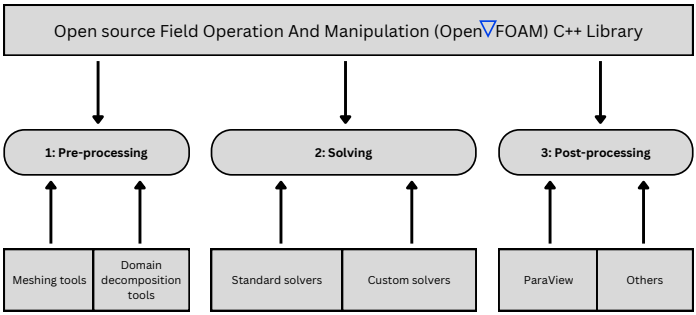


Figure 2.1. Overview of the OpenFOAM structure.

compared to their creation from scratch. Moreover, OpenFOAM includes third party softwares packages for some complementary tasks to the processing of the data. These utilities help the user to manipulate data with pre-processing (as changing the meshes through refinement and domain decomposition as well as modifying the initial setup of each case) and data post-processing (as data visualization through ParaView and plotting results).
Figure 2.2 shows the basic directory structure of an OpenFOAM standard case, including the essential files required to execute an application:

- ***0* directory**: this is the first of the time directories required by the solver to start the simulation. In this directory, the initial condition and the boundary conditions for each physical quantity of the simulation (e.g. pressure (p), velocity (U) and temperature (T)) are specified.

- ***constant* directory**: this folder contains the "polyMesh" folder where all the mesh data and the simulation boundary conditions are stored. It also contains all the properties needed to the solver, for instance the "thermophysicalProperties" file that specifies the thermodynamic and physical properties of the fluid or the "turbulenceProperties" file with the turbulence model used for the simulation.

- ***system* directory**: this folder contains the "controlDict" file where it is possible to define the settings of the simulation, for instance the used solver, the simulation time and the time-step. The methods for the discretization of the solved equations are specified in the "fvSchemes" file, while the solvers for the discretized equations are specified in the "fvSolutions" file. A special mention goes to the "*blockMesh*" utility command which creates parametric meshes with grading and curved edges. The mesh is generated from a dictionary file named blockMeshDict located in the *system* directory. "*blockMesh*" reads this dictionary, generates the mesh and writes out the mesh data files (boundary, faces, neighbour, owner and points) in the "polyMesh" folder, inside "constant". Moreover, depending on the case problem, other files can be included in the system directory, for instance the "decomposeParDict" file to decompose the domain simulation into several processors such to run the simulation in parallel.

For the sake of completeness, we will describe the type of files inside the "polyMesh" folder. As been said above, the polyMesh subdirectory contains the followin files:

- **boundary**: the file boundary lists the boundaries of the domain, with the faces f each boundary type referred to as a patch and assigned a name. The type of each boundary patch (type) is declared along with its number of faces (nFaces) and the starting face (startFace),which refers to the index of the first face in the list;

- **faces**: the file faces represents a list of faces, with each face described by a list of indices to vertices in the points list where the first entry in the list represents face 0, the second entry represents face 1 and so on;

- **neighbour**: it is a list of neighbour cell labels. The number of neighbours is basically equal the number of interior faces;

- **owner**: the file owner is a list in which the owner of faces are stored. The position of the owner in the list refers to the face it belongs to. The number of owners is equal to the total number of faces (interior + boundary faces);

- **points**: the file points is a list of vector denoting the cell vertices, with vertex 0 being the first vector in the list, vertex 1 the second vector, etc.

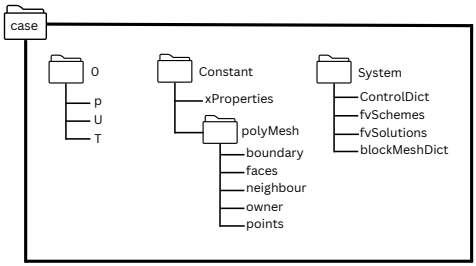


Figure 2.2. Case directory structure.

OpenFOAM can handle continuum mechanics problems resolving Partial Differencial Equations (PDEs) using the Finite Volume Method over arbitrary polyhedric meshes.

2.3 The Finite Volume Method

The Finite Volume Method (FVM)[12] is a numerical technique that transforms the partial differential equations representing conservation laws over differential volumes into discrete algebraic equations over finite volumes (or elements or cells). It gains popularity in CFD for being highly flexible as a discretization method. Even though it was preceded for many years by the Finite Difference (FD)[7] and Finite Element Method (FEM)[4], the FVM played a defining role in advancing the simulation of fluid flow problems and associated transport phenomena. This prominence arose from the groundbreaking work of the Computational Fluid Dynamics (CFD) group at Imperial College in the early 70s, led by Professor Spalding. Influential contributors such as Gosman [15] and Runchal and Wolfshtein [35] among others, made significant strides in developing methods that shaped the field. The FVM used in OpenFOAM follows the cell-centered logic, which is currently the most popular type of variable arrangement. This practice stores the quantities and the variables at the centroids of grid cells. More precisely, the solution domain is divided into a finite number of adjoining control volumes (V_p), and conservation equations are applied individually to each V_p. The computational node P is positioned at the centroid of each control volume, where variable

values are calculated. To estimate values at the surfaces of each V_p, interpolation is used to express these in terms of the nodal values located at the control volumes centers. Another way of looking at this is that FVM uses a volume integral formulation of the problem with a finite partitioning set of volumes to discretize the equations. The finite volume methods flexibility allows it to work with any type of grid, making it ideal for handling complex geometries. Additionally, the FVM ensures quantities conservation by construction, as long as the surface integrals, which represent convective and diffusive fluxes, are consistent across shared V_p boundaries. This means that this method is inherently designed to conserve key physical quantities—such as mass, momentum, and energy—within the solution domain. This conservation property is built into the method itself, ensuring that the total amount of a conserved quantity entering or leaving a control volume (CV) equals the net accumulation or depletion of that quantity within the CV.
Here we consider the general transport equation in its integral form:

$$\int_{V_p} \frac{\partial \rho\phi}{\partial t}\,\mathrm{d}V + \int_{V_p} \nabla\cdot(\rho u\phi)\,\mathrm{d}V - \int_{V_p} \nabla\cdot(\rho\Gamma_\phi\nabla\phi)\,\mathrm{d}V = \int_{V_p} S_\phi(\phi)\,\mathrm{d}V \tag{2.5}$$

The first term of the equation 2.5 is the integral over the volume V_P of the temporal derivative of the product of the density ρ and the generic transported quantity ϕ, the second term is the integral of the convective term (that is the product of the density ρ, the velocity field u and the transported quantity ϕ), the third is the integral of the diffusion term and the last is the integral of the source term. Now, let us recall the Gauss theorem which states that the outward flux of a vector field through a closed surface is equal to the volume integral of the divergence over the region inside the surface:

$$\int_V \nabla\cdot u\,\mathrm{d}V = \int_{\partial V} u\cdot n\,\mathrm{d}S \tag{2.6}$$

where ∂V is a closed surface bounding the generic control volume V and $\mathrm{d}S$ represents an infinitesimal surface element with associated normal $\hat{n}$ pointing outwards of the surface ∂V.
If we use the Gauss theorem to convert the volume integrals into surface integrals, that is using equation 2.6 into equation 2.5, we obtain the following equation

$$\frac{\partial}{\partial t}\int_{V_p}(\rho\phi)\,\mathrm{d}V+\int_{\partial V_p}(\rho u\phi)\cdot n\,\mathrm{d}S-\int_{V_p}(\rho\Gamma_\phi\nabla\phi)\cdot n\,\mathrm{d}S=\int_{V_p}S_\phi(\phi)\,\mathrm{d}V \tag{2.7}$$

Now we can take a step further and move from the continuous domain to the discrete domain for each term.
For the convective term we have:

$$\int_{\partial V_p}(\rho u\phi)\cdot n\,\mathrm{d}S=\sum_f\int_f(\rho u\phi)_f\cdot n\,dS\simeq\sum_f(\overline{\rho u\phi})_f\cdot n\,S_f=\sum_f(\rho u\phi)_f\cdot n\,S_f \tag{2.8}$$

For the diffusive term we have:

$$\begin{gathered}\int_{\partial V_p}(\rho\Gamma_\phi\nabla\phi)\cdot n\,\mathrm{d}S=\\ \sum_f\int_f(\rho\Gamma_\phi\nabla\phi)_f\cdot n\,\mathrm{d}S\simeq\sum_f(\overline{\rho\Gamma_\phi\nabla\phi})_f\cdot n\,S_f=\\ \sum_f(\rho\Gamma_\phi\nabla\phi)_f\cdot\hat{n}\,S_f\end{gathered} \tag{2.9}$$

where, in the third term of both the convective and diffusive parts, we have approximated the integrant by means of the mid point rule, which is second order accurate.
For the source term we have:

$$\int_{V_p}(\phi)S_\phi\,\mathrm{d}V=S_cV_P+S_pV_P\phi_P \tag{2.10}$$

This approximation is exact if S_ϕ is either constant or varies linearly with the control volume; otherwise is second order accurate. S_c is the constant part of the source term and S_P is the non-linear part.
Using equations 2.8, 2.9 and 2.10 into equation 2.7, we obtain the following semi-discrete transport equation:

$$\frac{\partial}{\partial t}\int_{V_p}(\rho\phi)\,dV+\sum_f(\rho u\phi)_f\cdot n\,S_f-\sum_f(\rho\Gamma_\phi\nabla\phi)_f\cdot n\,S_f=S_cV_P+S_pV_P\phi_P \tag{2.11}$$

where $(\rho u \phi)_f \cdot n \; S_f = F^C$ is the convective flux and $(\rho \Gamma_\phi \nabla \phi)_f \cdot n \; S_f = F^D$is the diffusive flux. All of this is necessary to transform the governing partial differential equations into a set of algebraic equations, one for each element in the computational domain. These algebraic equations are then assembled into a global matrix and vectors that can be expressed in the form

$$Ax = b \tag{2.12}$$

where A is the $N \times N$ coefficient matrix, x is the unknown N vector variable which is defined at each interior element and at the boundary of the computational domain, and b is the vector of dimension N of the known variables. One of the main advantages of the Finite Volume Method is that the techniques to solve the algebraic system of equations are independent of the discretization method, and represent the various trajectories that can be followed to obtain a solution. It is important to recall that the variables are computed and stored at the centroid P of the control volumes. This means that the face values in the convective and diffusive fluxes have to be computed by some form of interpolation from the centroid of the control volume at both sides of the face f.

2.4 Interpolation Schemes

Interpolation schemes are numerical methods used to estimate values at intermediate points within a discrete set of data or across a continuous domain. Interpolation methods range from simple approaches like linear and polynomial interpolation to more complex schemes designed to handle challenges like sharp gradients or discontinuities in data, which are common in physical simulations and engineering applications. Employing the Finite Volume Method (FVM), the partial differential equations are transformed into algebraic equations by integrating over discrete control volumes in the computational domain. This transformation requires both spatial and temporal discretization to approximate the solution within each control volume and over successive time steps.

2.4.1 Spatial Discretization

Spatial discretization is a critical component of the Finite Volume Method (FVM) when solving the Navier-Stokes equations, as it directly influences the accuracy, stability, and efficiency of the solution. In FVM, spatial discretization involves dividing the computational domain into a finite number of control volumes and approximating flow quantities and their derivatives, such as velocity and pressure, at the boundaries of these control volumes. When a the domain is two dimensional (2D) the boundaries of the control volumes is the edge of the cell and when it is three dimensional (3D) the boundary is the face of the control volume. This process enables the transformation of the continuous partial differential equations into discrete algebraic equations, which are then solved numerically.
For high-speed flows, effective spatial discretization is essential. In such flows, sharp gradients in velocity, temperature, and pressure are common, especially near shock waves, boundary layers, and regions of rapid compression and expansion. Accurately capturing these gradients without introducing numerical errors, such as diffusion oscillations, is essential for a realistic representation of the flow field. As discussed in the review on numerical methods for high-speed flows (Pirozzoli [33]), the interpolation schemes chosen for spatial discretization play a decisive role in balancing stability and accuracy under these challenging conditions.
In the context of the Navier-Stokes equations, spatial discretization must be applied to three terms: convective, diffusive, and gradient term. Each one requires distinct handling due to the different physical phenomena they represent. In the following paragraphs each term will be described.

Convective Terms

Convective terms model the transport of fluid properties, such as scalar quantities, momentum or energy, because they model the bulk motion of the flow. These terms are particularly dominant in high-speed flows, where their nonlinear nature often leads to steep gradients and discontinuities, such as shocks and expansions. Accurate resolution of these gradients is critical for capturing the essential features of compressible flows, but it poses significant numerical challenges. In fact, if the choosen discretization results inadequate, it can lead to excessive numerical diffusion with a consequent unphysical oscillations result

that compromise solution fidelity. Several discretization schemes have been developed in the past decades and they are currently employed for convective terms. For instance, the *Upwind Scheme*, which selects values from the upstream cell, is robust and stable, making it a popular choice for high-speed flows. However, its inherent numerical diffusion can obscure sharp gradients, making it less suitable for applications requiring high accuracy. In contrast, the *Central Differencing Scheme* (CDS) achieves higher accuracy by averaging values from adjacent cells. While effective in resolving smooth gradients, CDS is prone to numerical oscillations in flows dominated by convection, limiting its applicability in scenarios involving strong discontinuities.

To address these limitations, higher-order schemes, such as *QUICK* (Quadratic Upstream Interpolation for Convective Kinematics)[21] and *MUSCL* (Monotonic Upstream-Centered Schemes for Conservation Laws)[44], have been developed. These methods incorporate information from multiple neighboring cells to achieve greater accuracy and reduce numerical diffusion. QUICK is particularly effective in capturing smooth gradients, while MUSCL, with its flux-limiting approach, is better suited for problems with shocks and steep gradients. Another robust and widely used numerical scheme in CFD is the Advection Upwind Splitting Method (AUSM)[24], used for solving compressible flow problems. More precisely, it exist a rich AUSM family with integrating enhancements that improve accuracy, stability, and efficiency, particularly in the presence of shock waves, contact discontinuities, and low-Mach-number flows. For example $AUSM^+ - up$ achieves this by employing a blend of pressure-based and velocity-based flux splitting techniques while incorporating additional terms to stabilize pressure and velocity interactions. These refinements make $AUSM^+ - up$ especially suitable for a broad spectrum of aerospace applications, from high-speed aerodynamics to propulsion system modeling, where precision in capturing complex flow physics is critical. The AUSM formulation is described herein in section 2.4.3. The choice of the most suitable scheme in high-speed flows is a key aspect as it must balance accuracy and stability to handle the complex interplay of sharp gradients and nonlinearities.

The development of hybrid and adaptive schemes, which dynamically combine the strengths of multiple approaches, is an active area of research. These methods aim to achieve a balance between numerical diffusion and oscillations, making them particularly useful in simulations involving shocks, turbulence, and other complex flow features.

Diffusive Terms

Diffusion terms represent molecular diffusion processes, including viscosity and thermal conduction, which are governed by the physical properties of the fluid. Unlike convective terms, diffusion terms are generally linear, making their numerical treatment more straightforward. They are typically discretized using the *Central Differencing Scheme* (CDS), which provides second-order accuracy on uniform grids. The symmetry and non-oscillatory nature of central differencing ensure stable and accurate approximation of diffusive fluxes, making it a preferred choice for the Finite Volume Method (FVM) employed in solvers such as OpenFOAM.
Although diffusion effects are often small compared to convection in high-speed flows, their accurate representation is critical in regions where viscous effects dominate, such as boundary layers and shock-boundary layer interactions. In these areas, diffusion terms govern the transfer of momentum and energy at small scales, influencing key flow characteristics like skin friction and heat transfer. Proper discretization of these terms is essential for capturing the dynamics of shear layers, turbulence, and thermal gradients.
Recent advances in adaptive and high-order schemes have further improved the treatment of diffusion terms, especially in simulations involving non-uniform grids or complex geometries. These methods help mitigate errors arising from grid irregularities while preserving accuracy and computational efficiency, ensuring reliable results in high-fidelity simulations.

Gradient Terms

Gradient terms, such as those arising in pressure gradient calculations, play a crucial role in fluid dynamics as they govern the acceleration and deceleration of the flow. Accurate computation of these terms is essential for capturing the dynamics of pressure-driven flows, particularly in regions with sharp pressure changes or discontinuities, such as shock waves or boundary layer separations. In computational fluid dynamics, gradient terms are typically discretized using *linear central differencing*, a scheme that provides second-order accuracy on structured grids. This method is widely adopted in solvers like OpenFOAM due to its simplicity, stability, and effectiveness in regions of smooth flow.
While *linear central differencing* is effective for smooth flows, its re-

liance on grid uniformity can pose challenges in complex geometries or highly unstructured grids, where the accuracy of gradient calculations may deteriorate. To address these issues, advanced methods such as *least squares gradient reconstruction* and *weighted averaging techniques* have been developed. These approaches improve robustness and accuracy in non-uniform meshes, making them suitable for high-fidelity simulations in complex domains.
In high-speed flows, the accurate representation of pressure gradients is particularly critical, as errors in these terms can propagate through the solution, impacting the prediction of shock structures, expansion fans, and other key flow features. Therefore, careful selection and implementation of gradient discretization schemes are integral to achieving reliable and precise CFD results.

2.4.2 Flux Splitting Schemes

The story of flux splitting schemes begins in the late 1970s, when researchers were grappling with the challenge of accurately capturing shock waves and other discontinuities in compressible flow simulations. The key insight was that by splitting the flux terms into positive and negative components based on wave propagation directions, one could better handle these challenging flow features.The Godunov scheme, introduced by Sergei Godunov in 1959, represented a revolutionary approach to numerical fluid dynamics. This method solves the exact or approximate Riemann problem at cell interfaces, considering the flow as a series of interacting waves. The scheme's ability to naturally capture shock waves and maintain conservation properties made it a cornerstone of modern CFD, though its computational cost was initially seen as a drawback. Building on these foundations, the Total Variation Diminishing (TVD) scheme emerged in 1983 through the work of Harten et al. [16]. TVD schemes introduced the crucial concept of limiting the numerical solution's total variation to prevent spurious oscillations near discontinuities. This advancement addressed a key limitation of earlier high-order methods which often produced unphysical oscillations near shock waves. The TVD property ensures that the numerical solution remains stable and physically meaningful, making these schemes particularly valuable for compressible flow calculations. One further advancement in the flux splitting domain is the development of the Advection Upwind Splitting Method (AUSM) and its subsequent iterations: AUSM+ and AUSM+-up. The AUSM (Advection Upstream Splitting Method) scheme, developed by Liou and

Steffen [24] in 1993, represents a more sophisticated approach to flux splitting. It cleverly separates the convective and pressure terms in the flux vector, treating them differently based on their physical nature. The scheme combines the accuracy of flux-difference splitting methods like Godunov's with the computational efficiency of flux-vector splitting approaches. What makes AUSM particularly noteworthy is its ability to accurately capture both inviscid and viscous flow features while maintaining excellent robustness and efficiency. These three schemes are interconnected through their progressive refinement of flux splitting concepts. While Godunov's scheme laid the theoretical groundwork with its wave-based approach, TVD schemes added the crucial property of solution stability, and AUSM brought everything together with a physically motivated splitting that balanced accuracy and computational efficiency. Together, they represent key milestones in the development of modern computational fluid dynamics methods.

This section provides a comprehensive overview of these flux splitting schemes, their numerical algorithms, and their significance in addressing the challenges of all-speed flows.

Upwind methods

The upwinding approach, widely adopted within the gas dynamics community, is founded on the principle that the solutions of the Euler equations propagate along characteristic lines. Consequently, a stable numerical method must ensure that information is propagated in the same characteristic directions (Moretti [27]). Within the finite difference (FD) framework, the conventional methodology, known as flux vector splitting (Steger and Warming [43]), involves decomposing the flux function into positive and negative components:

$$f(u) = f^+(u) + f^-(u), \tag{2.13}$$

where $f^+(u)$ and $f^-(u)$ correspond to non-negative and non-positive propagation velocities, respectively, i.e.,

$$\frac{\mathrm{d}f^+}{\mathrm{d}u} \geq 0 \quad \text{and} \quad \frac{\mathrm{d}f^-}{\mathrm{d}u} \leq 0. \tag{2.14}$$

These components are discretized using left- and right-biased approximations to maintain linear stability.

In the finite volume (FV) framework, upwinding is typically achieved via the flux difference splitting approach, also referred to as the Godunov method. Considering the one-dimensional (1D) scalar conservation law:

$$\frac{\partial u}{\partial t} + a(u)\frac{\partial u}{\partial x} = 0 \tag{2.15}$$

which is used as a prototype for the development of numerical methods for hyperbolic equations, we can derive its FV semi-discretization form as follows:

$$\frac{\partial \overline{u}_j}{\partial t} = \frac{1}{h}(f(u_{j+1/2}) - f(u_{j-1/2})), \tag{2.16}$$

where $\overline{u}_j(t) = 1/h \int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t)dx$ is the spacial average of the approximate solution over the cell $I_j = (x_{j-1/2}, x_{j+1/2})$, a suitable reconstruction operator is used to determine approximate left and right states at the cell interfaces, $u^{\pm}_{j+1/2}$, and the interface flux $f(u_{j+1/2})$ is replaced with the numerical flux resulting from an exact (or approximate) Riemann solver, formally $\hat{f}_{j+1/2} = R(u^-_{j+1/2}, u^+_{j+1/2})$. For example, Roe's approximate Riemann solver prescribes

$$\begin{aligned} R(u^-_{j+1/2}, u^+_{j+1/2}) = \frac{1}{2}(f(u^-_{j+1/2}) + f(u^+_{j+1/2})) + \\ - \frac{|a_{j+1/2}|}{2}(u^+_{j+1/2} - u^-_{j+1/2}) \end{aligned} \tag{2.17}$$

where $a_{j+1/2}$ is the characteristic speed associated with the intermediate state and defined as

$$a_{j+1/2} = (f(u^+_{j+1/2}) - f(u^-_{j+1/2}))/(u^+_{j+1/2} - u^-_{j+1/2}). \tag{2.18}$$

In Godunov's original formulation, piece-wise constant reconstructions were assumed, leading to the general first-order flux

$$\hat{f}_{j+1/2} = R(\overline{u}_j, \overline{u}_{j+1}) \tag{2.19}$$

Upwinding has the main effect of damping the Fourier modes with

the highest supported wave numbers, with a subsequent stabilizing effect on the numerical solution.

2.4.3 The AUSM Schemes

The AUSM (Advection Upstream Splitting Method) scheme is a family of numerical schemes used in computational fluid dynamics (CFD) to solve compressible flow problems. Developed to enhance the accuracy and robustness of numerical simulations, the AUSM scheme efficiently handles shock waves, contact discontinuities, and smooth flow regions by carefully splitting the convective and pressure terms in the flux computation. Over time, various improved versions, such as AUSM+, AUSM+-up, and others, have been introduced to address stability and accuracy issues in high-speed flows. These schemes are widely applied in aerospace, automotive, and energy industries for simulating supersonic and transonic flows with high fidelity.

The AUSM Scheme

The original AUSM scheme, introduced by Liou and Steffen [24] in their seminal paper *"A New Flux Splitting Scheme"*, marked a significant departure from earlier flux vector and flux difference splitting methods. The key innovation of AUSM lies in its decomposition of fluxes into convective and pressure components, each treated separately to align with their respective physical characteristics.

Numerical Formulation

Considering the two-dimensional system of Euler equation for perfect gas:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{2.20}$$

where the velocity field is

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \tag{2.21}$$

the inviscid fluxes are

$$F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, \; G = \begin{pmatrix} \rho u \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{pmatrix}, \tag{2.22}$$

and the specific total energy is $E = e + 1/2(u^2 + v2) = H - p/\rho$. The AUSM scheme splits the inviscid flux F into two components:

$$\mathbf{F} = \mathbf{F^c} + \mathbf{F^p} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho H \end{pmatrix} u + \begin{pmatrix} 0 \\ p \\ 0 \\ 0 \end{pmatrix}, \tag{2.23}$$

where F_c represents the convective flux, and F_p represents the pressure flux. The convective terms can now be considered as passive scalar quantities convected by a suitably defined velocity u at the cell interface. On the other hand, the pressure flux terms are governed by the acustic wave speeds.The two components are discretized separately and at an interface $L < \dfrac{1}{2} < R$, the convective terms can be written as function of the Mach number M, ensuring upwinding to handle advection accurately:

$$F^c_{1/2} = M_{1/2} \begin{pmatrix} \rho a \\ \rho au \\ \rho av \\ \rho aH \end{pmatrix}_{L/R}, \tag{2.24}$$

where ($M_{1/2} \simeq M^+_L + M^-_R$) is the value of the split Mach number at the interface (red line in figure 2.3) and a is the speed of sound. Various ways of defining the split Mach number exist, for instance the Van Leer[45] splitting:

$$M^{\pm} = \begin{cases} \pm\dfrac{1}{4}(M \pm 1)^2 & if \;\; |M| \leq 1; \\ \dfrac{1}{2}(M \pm |M|), & if \;\; otherwise \end{cases} \tag{2.25}$$

where M^+_L is the value of the left contribution of the Mach number and M^-_R is the value of the right contribution of the Mach number. Also

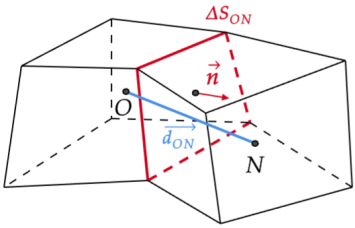


Figure 2.3. Computational cell for numerical flux evaluation. ΔS_{ON} represents the interface between the owner cell O and the neighboring cell N, with $\vec{n}$ as the outward normal for O and $\vec{d_{ON}}$ the distance between the cell centers O and N.

$$(\cdot)_{L/R} = \begin{cases} (\cdot)_L & if \;\; M_L^+ \geq M_R^- \\ (\cdot)_R & if \;\; M_L^+ < M_R^- \end{cases} \tag{2.26}$$

where L and R are the Left and Right side of two adjacent cells respectively, as in figure 2.3.
The pressure flux is treated symmetrically to ensure accurate shock resolution and smooth transitions:

$$F_p = p_{1/2} = p_L^+ + p_R^-. \tag{2.27}$$

The AUSM scheme is simple and computationally efficient, providing accurate shock capturing and robust performance for subsonic and supersonic flows, sometimes with better results than the ones obtained with the Roe scheme. However, it showed limitations in maintaining stability and accuracy for low-Mach-number flows.

The AUSM+ Scheme

To address the deficiencies of AUSM, particularly removing postshock overshoot and a glitch in the slowly moving shock problem, Liou [22] introduced the $AUSM+$ scheme in the follow-up paper. This iteration improved robustness and extended the method's applicability to a broader range of flow conditions. In fact, the scheme is capable of exactly capturing stationary shocks and contact discontinuities, it ensures that density remains positive, which is crucial for handling

strong rarefaction and near-vacuum flows, and it significantly reduces the "carbuncle" issue that affects the Roe scheme and previous AUSM implementations in blunt-body simulations. The AUSM+ scheme improves the splitting approach by unifying the Mach number and velocity formulations while maintaining a simple structure, making it easily adaptable to other hyperbolic systems.

Algorithm

To implement the AUSM+ to a basic central-differencing code, it only requires the key interface quantities $(m_{j+1/2}, p_{j+1/2})$ and unified numerical speed of sound $a_{j+1/2}$. The Mach number is split to minimize numerical dissipation at low Mach numbers.

$$\begin{aligned} 2m_{j+1/2} = (M_j + M_{j+1}) - (\mathscr{M}^+_{j+1} - \mathscr{M}^-_{j+1}) + \\ +(\mathscr{M}^+_j - \mathscr{M}^-_j) - \delta m_{j+1/2}, \end{aligned} \tag{2.28}$$

where

$$|\mathscr{M}|(M) = \mathscr{M}^+(M) - \mathscr{M}^-(M), \tag{2.29}$$

$$\delta m = |\mathscr{M}|(M - j + 1) - |\mathscr{M}|(M_j) \tag{2.30}$$

and

$$|\mathscr{M}|(M) = \begin{cases} |M|, & |M| \geq 1 \\ \dfrac{1}{2}(M^2+1) + 2\beta(M^2-1)^2, & otherwise \end{cases} \tag{2.31}$$

Since $|\mathscr{M}| > 0 \forall M$, the absolute sign to $\mathscr{M}$ make sense. On the other hand $\Delta|\mathscr{P}|$ can be either positive or negative, depending on the sign of M:

$$|\mathscr{P}|(M) = \begin{cases} sign(M), & |M| \geq 1 \\ \dfrac{M}{2}(3 - M^2 + 4\alpha(M^2-1)^2), & otherwise \end{cases} \tag{2.32}$$

Here, the pressure flux is modified to ensure smooth transitions between subsonic and supersonic regimes. The pressure flux incorporates a blending function to adapt dynamically

$$2p_{j+1/2} = (p_j + p_{j+1}) - \delta p_{j+1/2} \tag{2.33}$$

where

$$\delta p_{j+1/2} = \Delta\mathscr{P}(M_{j+1})p_{j+1} - \Delta\mathscr{P}(M_j)p_j \tag{2.34}$$

and

$$\Delta\mathscr{P}(M) = \mathscr{P}^+(M) - \mathscr{P}^-(M) \tag{2.35}$$

Once the quantities $(\delta p_{j+1/2}, \delta m_{j+1/2})$ have been defined, the dissipative terms derived from the AUSM+ are ready to be added. Rewriting the AUSM+ flux in terms of the central-difference (CD) formula yields,

$$f_{j+1/2}^{AUSM^+} = f_{j+1/2}^{CD} + f_{j+1/2}^{vis} \tag{2.36}$$

where

$$f_{j+1/2}^{vis} = f_{j+1/2}^{C} + f_{j+1/2}^{P} \tag{2.37}$$

or, explicitly, the convection term is

$$\begin{aligned} f_{j+1/2}^{C} = -\frac{a_{j+1/2}}{2}\Big[\Big(\frac{1}{2}\delta m_{j+1/2} - |m_{j+1/2}|\Big)\Phi_j + \\ + \Big(\frac{1}{2}\delta m_{j+1/2} + |m_{j+1/2}|\Big)\Phi_{j+1}\Big] \end{aligned} \tag{2.38}$$

and the pressure term is

$$f_{j+1/2}^{P} = -\frac{1}{2}\begin{pmatrix} 0 \\ \delta p_{j+1/2} \\ 0 \end{pmatrix}. \tag{2.39}$$

This scheme improved stability and accuracy at low Mach numbers, enhanced shock and contact discontinuity resolution, and reduced dissipation for smooth flow regions.

The AUSM+-up Scheme

Recognizing the need for a unified framework capable of handling flows across all speeds, which could not be achieved with AUSM+, in 2006 Liou [23] proposed the AUSM+-up scheme in *"A Sequel to AUSM, Part II: AUSM+-up for All Speeds"*. As for its predecessors, this method integrates pressure-based and velocity-based splitting strategies, with more precise corrections that allowed the achieving of superior performances. In fact, the previous AUSM+ scheme had two critical deficiencies that affect compressible flow solvers, particularly when applied to low-speed flow predictions. These deficiencies manifest as two distinct yet interconnected problems: an intrinsic convergence instability and a fundamental inaccuracy in solution generation. The first issue originates at the continuum level, fundamentally tied to the governing equations' mathematical structure. This convergence challenge exists independently of the spatial discretization scheme employed, representing a deep-rooted numerical complexity. The second deficiency, however, is directly linked to the flux numerical scheme itself, offering a more tractable avenue for methodological intervention. Previous computational approaches, despite their practical successes, revealed significant limitations. The pressure split function exhibited a critical discontinuity at zero Mach number, necessitating arbitrary cut-off parameters that artificially constrained the method's applicability. Existing preconditioners relied on preset Mach number limitations, which fundamentally restricted the method's universal potential. The core objectives of the Liou's latest article are ambitious yet precise: to create a flux scheme that ensures convergence rates independent of Mach numbers in low-speed flows, maintains at least equivalent performance in other speed regimes, and preserves solution accuracy across the entire spectrum of flow conditions.

Algorithm

The AUSM+-up scheme introduces two critical modifications:

- The pressure flux incorporates an additional dissipation term proportional to the pressure difference to stabilize low-Mach-number flows.
- The convective flux is enhanced by adding a pressure-difference-driven velocity correction, ensuring robustness across all speed regimes.

The Mach number and speed of sound at the interface are defined as:

$$M_{L/R} = \frac{u_{L/R}}{a_{1/2}}, \tag{2.40}$$

where $a_{1/2}$ can be an average of a_L and a_R. For multi-dimensional flows, $u = \mathbf{V} \cdot \mathbf{n}$, with $\mathbf{n}$ as the unit normal vector of the cell face under consideration.
The following auxiliary quantities are computed:

$$\bar{M}^2 = \frac{u_L^2 + u_R^2}{2a_{1/2}^2}, \tag{2.41}$$

$$M_o^2 = \min(1, \max(\bar{M}^2, M_1^2)), \tag{2.42}$$

$$f_a(M_o) = M_o(2 - M_o), \tag{2.43}$$

where $f_a(M_o)$ is a scaling factor.

Then, we can compute the mass flux and pressure fluxes as in 2.44 and 2.46

$$\dot{m}_{1/2} = a_{1/2} M_{1/2} \begin{cases} q_L & \text{if } M_{1/2} > 0, \\ q_R & \text{otherwise,} \end{cases} \tag{2.44}$$

where $M_{1/2}$ is defined as:

$$M_{1/2} = M_+^{(4)}(M_L) + M_-^{(4)}(M_R) - \frac{K_p}{f_a} \max(1 - r\bar{M}^2, 0) \frac{p_R - p_L}{q_{1/2} a_{1/2}^2}, \tag{2.45}$$

with $q_{1/2} = \dfrac{q_L + q_R}{2}$.

$$\begin{aligned} p_{1/2} = P_+^{(5)}(M_L) p_L + P_-^{(5)}(M_R) p_R + \\ -K_u P_+^{(5)}(M_L) P_-^{(5)}(M_R)(q_L + q_R) f_a a_{1/2} (u_R - u_L), \end{aligned} \tag{2.46}$$

where $P_+^{(5)}$ and $P_-^{(5)}$ are fifth-degree polynomial functions.

Finally, the total flux is given by:

$$F_{1/2} = \dot{m}_{1/2}\tilde{w}_{L/R} + p_{1/2}, \tag{2.47}$$

where $\tilde{w}_{L/R}$ represents the convected variables, selected upwinded based on the sign of $\dot{m}_{1/2}$.

With these modifications, the AUSM+-up scheme achieves some advantages:

- Unified handling of incompressible, subsonic, transonic, and supersonic flows.
- Improved accuracy for steady and unsteady low-Mach-number flows.
- Robust shock and contact discontinuity resolution.

Finally we can say that the AUSM family has profoundly impacted CFD, providing reliable tools for simulating a wide range of aerodynamic and thermodynamic problems. Applications span from aerospace engineering to turbomachinery, internal combustion engines, and atmospheric modeling. The AUSM+-up scheme, in particular, has become a standard choice for many high-fidelity solvers, owing to its ability to handle diverse flow phenomena without compromising stability or accuracy.

2.4.4 Shock Capturing Schemes

The shock-capturing method applies a uniform discretization scheme across all points and achieves stabilization by incorporating numerical dissipation, which prevents the emergence of Gibbs oscillations. In contrast, the shock-fitting method treats shock waves as true discontinuities, governed by their own algebraic equations and it employs the Rankine-Hugoniot relations as nonlinear boundary conditions to connect the states on either side of the discontinuity.

Classical methods

In the late 1950s, shock-capturing schemes have been designed starting from the features of the entropy solution of the model scalar conservation law 2.15, as follows:

- **Monotonicity preservation** which states that initially monotone data remain monotone for all times;
- **Total Variation Diminishing** (TVD) which in mathematical form states

$$TV(u(\cdot, t_2)) \leq TV(u(\cdot, t_1)), \quad \forall t_2 \geq t_1 \tag{2.48}$$

  where $TV(u(\cdot, t))$ denotes the total variation of u, as follows for a discrete sequence u_j

$$TV(u) = \sum_{j=1}^{n} |u_{j+1} - u_j| \tag{2.49}$$

- **Monotonicity**: we consider two solutions of equation 2.15, for instance $u(x,t)$, $v(x,t)$.
  The monotonicity property states that if $u(x,0) \leq v(x,0) \quad \forall(x)$ then

$$u(x,t) \leq v(x,t) \quad \forall(x,t) \tag{2.50}$$

Owing to the availability of rigorous theorems ensuring the convergence of TVD schemes to weak solutions and monotone schemes to entropy solutions, many shock-capturing implementations satisfying these conditions have appeared in the literature. Even though TVD schemes gained wide success in the 1980s, the main problem of monotone schemes is that they can achieve at best first-order accuracy with a conseguent loss of accuracy at both smooth and non-smooth extrema. This led researchers to pursue alternatives for constructing uniformly high-order accurate shock-capturing schemes. The Essentially NonOscillatory (ENO) schemes introduced by Harten et al. [16] address this issue by computing the numerical flux via a high-order reconstruction over an adaptively chosen stencil that minimizes interpolation across discontinuities, thus reducing Gibbs oscillations. ENO reconstructions of order r ensure that the increase in total variation is limited to $O(h^r)$, which keeps the total variation uniformly bounded. However, the free adaptation of stencils in ENO schemes can cause convergence problems and loss of accuracy. Building on these ideas, Weighted Essentially NonOscillatory (WENO) schemes were later introduced by Liu et al. [25]. These schemes construct a single high-order numerical flux as a convex linear combination of several

lower-order polynomial reconstructions over a set of staggered stencils. The weights of the stencils are chosen so that maximum formal accuracy is achieved in smooth regions while almost zero weight is given to reconstructions that cross a discontinuity. Subsequent research has focused on enhancing the performance of WENO schemes, especially by improving their behavior in smooth regions.

Hybrid Methods

This class of methods enhances a basic spectral-like scheme with shock-capturing capabilities by either locally switching to a conventional shock-capturing method or by selectively incorporating the dissipative component of a shock-capturing method acting as a nonlinear filter. A key role in this class of schemes is played by shock sensors that must be defined in such a way that numerical dissipation is effectively confined in shocked regions, so that it does not pollute smooth parts of the flow field. For instance, to analyze the interaction between an isotropic turbulent field and a planar shock wave, we could use a hybrid discretization of a high-order compact scheme and a high-order shock-capturing ENO scheme but, in this case, the shock location is approximately known, so the switch between the two schemes can be decided a priori. Adams and Shariff [1] initially explored a truly adaptive hybrid discretization, combining a baseline compact upwind scheme with a fifth-order ENO scheme. A simple switching mechanism, based on the local gradients of the flux vector components, was employed to identify critical cells. These cells were then surrounded by buffer cells on each side to preclude oscillations caused by the coupling of different properties schemes. In 2002, Pirozzoli [30] extended the approach by formulating a fully conservative scheme that blends a fifth-order compact upwind numerical flux with a seventh-order WENO flux, employing a switching mechanism determined by the local density gradient. Even though hybrid schemes are a valid alternative to the classical methods above, it appears that sometimes the coupling between two methods may give rise to an unstable system, thus requiring a deeper investigation. However, if the two schemes used in the coupling are both dissipative, then the coupled system is strongly stable, therefore endorsing the use of upwind shock-capturing schemes around discontinuities. The Jameson–Schmidt–Turkel (JST[19]) scheme, introduced in 1981, employs central differencing for spatial discretization and incorporates artificial dissipation terms to maintain numerical stability and accurately capture shock waves. The JST scheme com-

putes numerical fluxes using a central difference method with added artificial dissipation to ensure stability and prevent oscillations in shock-dominated flows. These components are controlled by sensors based on pressure variations. The scheme has also been reformulated as a total-variation-diminishing (TVD) scheme, aligning it with modern high-resolution methods. Suppose that the scalar conservation law is the one in equation 2.15 which can be approximated by the semidiscrete scheme

$$\Delta x \frac{\mathrm{d}v_j}{\mathrm{d}t} + \hat{f}_{(j+1/2)} - \hat{f}_{(j-1/2)} = 0 \tag{2.51}$$

where v_j represents the average value of u in cell j, and $f_{(j+1/2)}$ is the numerical flux across the interface separating cells j and $j+1$. Introducing artificial diffusion, the JST numerical flux can be defined as follows:

$$\hat{f}_{(j+1/2)} = \hat{f}^C_{(j+1/2)} - d_{(j+1/2)} \tag{2.52}$$

where the diffusive flux has the following form

$$\begin{aligned} d_{(j+1/2)} = \epsilon^2_{(j+1/2)} \Delta v_{(j+1/2)} + \\ -\epsilon^4_{(j+1/2)} (\Delta v_{(j+3/2)} - 2\Delta v_{(j+1/2)} + \Delta v_{(j-1/2)}) \end{aligned} \tag{2.53}$$

with $\Delta v_{(j+1/2)} = v_{j+1} - v_j$, which becomes:

$$d_{(j+1/2)} = \epsilon^2_{(j+1/2)} (v_{j+1} - v_j) - \epsilon^4_{(j+1/2)} (v_{j+2} - 3v_{j+1} + 3v_j - v_{j-1}) \tag{2.54}$$

Here, we can see that the dissipation term consists of a second-order component (to handle small oscillations) and a fourth-order component (to reduce excessive dissipation in smooth regions). Let $a_{(j+1/2)}$ be the numerically estimated wave speed

$$a_{j+1/2} = \begin{cases} \dfrac{f_{j+1} - f_j}{v_{j+1} - v_j}, & v_{j+1} \neq v_j \\ \left.\dfrac{\partial f}{\partial u}\right|_{u=v_j}, & v_{j+1} = v_j \end{cases} . \tag{2.55}$$

So the JST scheme is TVD whenever v_j and v_{j+1} is an extremum,

$$\epsilon^2_{(j+1/2)} \geq \frac{1}{2}|a_{(j+1/2)}|, \quad \epsilon^4_{(j+1/2)} = 0. \tag{2.56}$$

To construct coefficients $\epsilon^2_{(j+1/2)}$ and $\epsilon^4_{(j+1/2)}$ that satisfy the previous conditions, it is necessary to define the following function:

$$R(u, v) = \left| \frac{u - v}{|u| + |v|} \right|^q \tag{2.57}$$

where $q \geq 1$. Then, if u and v have opposite signs,

$$R(u, v) = 1. \tag{2.58}$$

Now, set

$$\begin{cases} \epsilon^2_{(j+1/2)} = \alpha_{(j+1/2)} Q_{j+1/2} \\ \epsilon^4_{(j+1/2)} = \beta_{(j+1/2)}(1 - Q_{j+1/2}) \end{cases} \tag{2.59}$$

where

$$Q_{j+1/2} = R(\Delta v_{(j+3/2)}, \Delta v_{(j-1/2)}). \tag{2.60}$$

Because $\Delta v_{(j+3/2)}$ and $\Delta v_{(j-1/2)}$ have opposite signs if either v_j or v_{j+1} is an extremum, the scheme will be a local extremum diminishing (LED) scheme if

$$\alpha_{(j+1/2)} \geq \frac{1}{2}|a_{(j+1/2)}|. \tag{2.61}$$

Typically,

$$\beta_{(j+1/2)} = k_4 |a_{(j+1/2)}| \tag{2.62}$$

where in the case of steady-state calculations, k_4 can be tuned to maximize the rate of convergence to a steady state. This realization of the JST scheme is actually an example of a symmetric TVD scheme. Overall, the formulation of the JST scheme was guided by a number of design principles such as:

- The scheme should be in conservation form to ensure satisfaction of the shock jump conditions, according to the theorem of Lax and Wendroff [20].

- There should be second-order accuracy in smooth regions of the flow.
- Shock waves should be captured without overshoots or oscillations, at least in the steady state (but overshoots during the transient phase would be tolerated).
- The steady state should be independent of the time evolution.
- The scheme should be stable when using variable local time steps at a fixed CFL number to accelerate convergence to a steady state.
- The discrete steady-state solution should have constant stagnation enthalpy, consistent with properties of the true steady-state solutions.

Moreover, the artificial flux of the Jameson-Schmidt-Turkel (JST) scheme in equation 2.63 was also used by Ducros et al. [9] to modify the original nonlinear filtering approach, initially introduced by Yee et al. [47] who designed one of the first low-dissipative, shock-capturing algorithm. In fact, Ducros replaced the Harten [16] switch, which is designed to be almost unity near shocks and almost zero in smooth parts of the flow, with the product of the Jameson sensor and another shock sensor, that is known as the "Ducros Sensor".

$$\psi_j = \frac{|v_{j+1} - 2v_j + v_{j-1}|}{|v_{j+1} + 2v_j + v_{j-1}|}, \quad 0 \leq \psi_j \leq 1. \tag{2.63}$$

Ducros Sensor The Ducros sensor, introduced by Ducros et al. [9], is a tool designed to distinguish between compressible shocks and other flow features such as vortices, thus aiding in the adaptive application of numerical dissipation. It is defined as:

$$\phi = \frac{(\nabla \cdot \mathbf{u})^2}{(\nabla \cdot \mathbf{u})^2 + (\nabla \times \mathbf{u})^2 + \epsilon}, \tag{2.64}$$

where $(\nabla \cdot \mathbf{u})$ is the flow divergence, $(\nabla \times \mathbf{u})$ is the vorticity magnitude, and ϵ is a small positive constant to prevent division by zero. The sensor takes values between 0 and 1, with values close to 1 indicating regions dominated by compression (shocks), and values close to 0 identifying vortex-dominated regions. This makes it particularly effective for hybrid and nonlinear filtering schemes, where dissipation needs to

be selectively introduced. Extensions of the Ducros sensor have been employed in direct numerical simulations and large-eddy simulations, as it can reduce the risk of incorrectly identifying vortices as shocks, ensuring a better distinction of flow dynamics. This enhances computational efficiency and accuracy in solving compressible turbulent flows with shocks. Later studies conducted by Garnier et al. [14] showed that the Ducros sensor is capable of distinguishing turbulent fluctuations from shocks better than the Harten [16] switch.

2.4.5 Temporal Discretization

Temporal discretization allows the solution to evolve over time, which is an essential process in unsteady simulations. In computational fluid dynamics, the choice of time discretization scheme plays a significant role in determining both accuracy and computational efficiency of the solution. In the following paragraphs, the main time discretization methods are described, analysing for each their stability, accuracy and computational cost properties.

Explicit Time Discretization

Explicit schemes are numerical methods in which the solution at the next time step is calculated directly from the known values at the current time step. A commonly used explicit method is the *forward Euler*[8] scheme, which approximates the time derivative as:

$$F(\phi^n) \approx \frac{\phi^{n+1} - \phi^n}{\Delta t}. \tag{2.65}$$

Here, ϕ^{n+1} represents the unknown solution at the next time step, ϕ^n is the known solution at the current time step and Δt is the time step size. So the unknown flux ϕ^{n+1} can be evaluated starting from equation 2.65 as follows:

$$\phi^{n+1} \approx \ \phi^n + F(\phi^n)\ \Delta t. \tag{2.66}$$

Starting from a fixed Δt and knowing the function $F(\phi^n$, the stored quantities of the known values at the current time ϕ^n can be used to evaluate the unknown values of the quantities at the next time ϕ^{n+1}. This makes explicit schemes straightforward to implement and computationally efficient. Explicit schemes are conditionally stable

and require the time step size Δt to satisfy specific stability criteria, such as the Courant-Friedrichs-Lewy (CFL) condition. For example, in advection-dominated problems, the CFL condition imposes the following constraint:

$$\Delta t \ \leq \ C_{max} \ \frac{\Delta x}{|\lambda_{\mathbf{u}}|}, \tag{2.67}$$

where Δx is the spatial grid resolution and $|\lambda_{\mathbf{u}}|$ is the characteristic velocity of the flow. Failure to meet this criterion may result in numerical instability. The value of C_{max} changes with the method used to solve the discretised equation, especially depending on whether the method is explicit or implicit. If an explicit solver is used then typically $C_{max} \leq 1$. Implicit solvers are usually less sensitive to numerical instability, so $C_{max} > 1$. The accuracy of explicit schemes depends on their formulation. In general, in order for this kind of schemes to be accurate the CFL require a very low Δt (e.g. for a dimensionless time, that would correspond to an order of magnitude of 10^{-6}). This requires large computation times to be sure the simulation has come to convergence. The *forward Euler* method is first-order accurate, meaning that its error decreases linearly with the time step size. Higher-order explicit schemes, such as *Runge-Kutta* methods, provide improved accuracy by incorporating additional intermediate steps within each time step. Explicit schemes have a low computational cost per time step compared to other schemes because they avoid solving coupled systems of equations. Instead, each time step involves simple arithmetic operations, which makes these methods suitable for problems where computational efficiency is a priority. Explicit schemes are particularly suitable for problems where the solution evolves rapidly, and small time steps are naturally required for accuracy. They are commonly employed in transient simulations dominated by advection or in problems involving wave propagation, where the small time scales ensure stability and precision.

Implicit Time Discretization

Implicit schemes are widely employed in CFD, particularly for simulations where numerical stability is a primary concern, such as in unsteady flow problems with large time steps or in systems involving strong coupling between physical processes. This kind of schemes provide a robust approach for solving time-dependent problems by

incorporating unknown future states into the formulation. This characteristic allows implicit methods to operate with larger time steps compared to explicit schemes, making them particularly advantageous for stiff problems commonly encountered in computational fluid dynamics. A widely used implicit time discretization is the *Backward Euler*[8] method, which approximates the time derivative as:

$$F(\phi^{n+1}) \approx \frac{\phi^{n+1} - \phi^n}{\Delta t}, \tag{2.68}$$

where ϕ^{n+1} is the unknown variable at the new time step, ϕ^n is the known value at the current time step, and Δt is the time step size.

This scheme is said to be implicit because the calculation of the unknown variable at the next time step ϕ^{n+1} is implicitly depending on itself:

$$\phi^{n+1} \approx \ \phi^n + F(\phi^{n+1})\ \Delta t. \tag{2.69}$$

It is now well known that the accuracy of implicit schemes depends on their formulation. The backward Euler method is first-order accurate and unconditionally stable for many linear problems. For scenarios demanding improved accuracy, the *Crank-Nicolson*[7] method - a second-order implicit scheme - can be employed. This method averages the time derivative between the current and next time steps, providing greater accuracy but sometimes introducing oscillations in highly nonlinear systems or when large time steps are used. Implicit schemes are unconditionally stable for many linear problems, allowing the use of larger time step sizes without the risk of numerical instability. This is particularly beneficial for stiff systems, such as high Reynolds number flows or strongly coupled multi-physics problems. Implicit schemes are also computationally expensive as they require solving a coupled system of algebraic equations at each time step. This cost can become significant for large-scale problems in which the use of efficient solvers and preconditioners to ensure feasibility are required. Despite their computational cost, their unconditional stability and ability to handle stiff equations make them a very powerful and are commonly used in many CFD software programs.

2.5 Interpolation schemes in OpenFOAM

In OpenFOAM, the selection of interpolation schemes for spatial and temporal terms is specified in the `fvSchemes` and `fvSolution` configuration files. The `fvSchemes` file determines the spatial interpolation schemes for divergence, gradient, and Laplacian terms, while `fvSolution` configures the temporal schemes and solvers for each variable. For high-speed flows, the selection of schemes is particularly critical, as these flows are sensitive to both numerical diffusion and oscillations. Typically, an initial solution might employ an upwind scheme for stability, which can then be refined with higher-order schemes, such as *Roe*[34], *TVD*[16] or *AUSM*[24], for improved accuracy. The Roe scheme is a popular approximate Riemann solver that linearizes the flux Jacobian to achieve high accuracy and ensure proper wave propagation by capturing discontinuities with minimal numerical dissipation. The Godunov scheme, on the other hand, is a fundamental first-order method that employs the exact solution of the Riemann problem to determine fluxes at cell interfaces, ensuring robust shock capturing, albeit with relatively high dissipation. To overcome the excessive numerical diffusion of first-order methods, Total Variation Diminishing (TVD) schemes were introduced, incorporating non-oscillatory high-order reconstructions such as MUSCL (Monotonic Upstream-Centered Schemes for Conservation Laws) to enhance solution accuracy while maintaining stability. The Advection Upstream Splitting Method (AUSM) family of schemes takes a different approach by splitting convective and pressure fluxes, offering superior accuracy in capturing compressible flow features like shocks and contact discontinuities, especially in transonic and supersonic regimes. Together, these methods highlight the evolution of upwind techniques, balancing accuracy, stability, and computational efficiency for complex flow problems.

2.5.1 OpenFOAM solvers

OpenFOAM is known for its flexibility and extensive range of solvers tailored to specific simulation needs. The solvers in OpenFOAM are broadly categorized into incompressible and compressible solvers, addressing fluid flow scenarios depending on the fluid's compressibility and the physical phenomena being modeled. Incompressible solvers are designed for flows where density variations are negligible, such as low-speed air or water flows. Examples include simpleFoam for

steady-state turbulent flows, pimpleFoam for transient simulations using the SIMPLE and PIMPLE algorithms, and icoFoam for laminar, transient flows. On the other hand, compressible solvers handle flows where changes in density due to pressure and temperature variations are significant, as in high-speed aerodynamics, supersonic flows and combustion processes. Examples include rhoSimpleFoam and rhoPimpleFoam for steady and transient compressible flows, respectively, and rhoCentralFoam, which leverages a central difference scheme for shock-dominated flows. Beyond these categories, OpenFOAM also includes specialized solvers for heat transfer, multiphase flows, chemical reactions, and electromagnetics. For instance, solvers like chtMultiRegionFoam enable conjugate heat transfer analysis, while interFoam and twoPhaseEulerFoam address multiphase flow problems. Each solver is tailored to specific applications, allowing users to simulate a wide range of industrial and research problems with high accuracy and computational efficiency. Furthermore, the modularity of OpenFOAM enables customization of solvers, providing unparalleled flexibility to engineers and researchers.

Simple, Piso and Pimple Methods

Developed in the early 1970s by Professor Brian Spalding and his student Suhas Patankar [29] at Imperial College London, the SIMPLE algorithm has become a cornerstone in CFD research and applications. Over the years, it has been widely adopted and extensively applied by researchers to address a broad range of fluid flow and heat transfer problems, solidifying its position as a key method in the numerical simulation of fluid dynamics. The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm is widely used in OpenFOAM for addressing the pressure-velocity coupling inherent in the Navier-Stokes equations. This coupling arises from the incompressibility condition, which requires that the velocity field remain divergence-free.

The SIMPLE algorithm is primarily designed for steady-state simulations: in fact it do not contain any time derivation in the equations. In this approach, the momentum equation is first solved using an initial guess for the pressure field to obtain an intermediate velocity field. Since this velocity field does not necessarily satisfy the continuity equation, a pressure correction equation is derived from the continuity condition. This correction is applied iteratively to adjust the pressure field and subsequently the velocity field until the solution converges.

The SIMPLE algorithm is computationally efficient for steady-state problems due to its iterative nature, but it can converge slowly for problems with strong nonlinearities or transient effects.
On the other hand, the PISO (Pressure Implicit with Splitting of Operators) algorithm[18] is also a method for solving the implicity discretised, time-dependent, Navier-Stokes equations, but it uses a non-iterative approach within each time step, which makes it computationally efficient for transient problems and it is better suited for unsteady simulations. As for the SIMPLE algorithm, the PISO algorithm begins with a solution of the momentum equation using an initial guess for the pressure field, leading to an intermediate velocity field. A pressure correction equation is then derived to ensure that the velocity field satisfies the continuity equation. After the pressure field is corrected, the velocity field is updated accordingly. Unlike SIMPLE, PISO may include additional corrector steps within a single time step to improve the accuracy of the solution. These corrector steps are optional and are determined by the specific requirements of the problem.
The primary difference between SIMPLE and PISO lies in their application and strategy. SIMPLE is iterative across all equations and is optimized for steady-state problems, where convergence to a final solution is required. On the other hand, PISO avoids iteration across time steps and is designed for transient problems, allowing larger time steps while maintaining numerical stability. Both methods are implemented in OpenFOAM with configurable parameters in the `fvSolution` dictionary, allowing users to adapt the algorithms to specific cases.
Lastly, the PIMPLE algorithm, is a hybrid scheme obtained from SIMPLE and PISO: it combines the iterative robustness of SIMPLE with the transient capabilities of PISO, making it versatile for both steady-state and transient problems. Within each time step, PIMPLE employs an outer iteration loop akin to SIMPLE, predicting the velocity field and applying pressure corrections until a specified convergence criterion is met. Simultaneously, it integrates the inner correction steps characteristic of PISO to achieve tighter pressure-velocity coupling. This dual-level approach allows PIMPLE to handle large time steps and complex flow conditions with improved stability, making it particularly effective for simulations involving turbulence or multiphase flows.
In the following sections there is a brief description of many incompressible and compressible solver available in OpenFOAM library and at at the bottom of the page, table 2.1 summarize all the solvers

described with their properties.

simpleFoam

`simpleFoam` is a steady-state, pressure-based solver for incompressible flows, using the SIMPLE algorithm. It is commonly used in industrial and aerodynamic applications where steady-state conditions are dominant.

pimpleFoam

`pimpleFoam` is a transient, pressure-based solver for incompressible flows, using the PIMPLE algorithm. It is robust for large time steps and high Courant number simulations.

pisoFoam

`pisoFoam` is a transient, pressure-based solver for incompressible flows, using the PISO algorithm. It is widely used for unsteady turbulent flow simulations.

rhoSimpleFoam

`rhoSimpleFoam` is a steady-state, pressure-based solver for compressible flows. It employs the SIMPLE algorithm, making it suitable for steady aerodynamics and heat transfer applications.

rhoPimpleFoam

`rhoPimpleFoam` is a transient, pressure-based solver for compressible flows. It uses the PIMPLE algorithm, providing stability for high Courant number simulations.

rhoPisoFoam

`rhoPisoFoam` is a transient, pressure-based solver for compressible flows, using the PISO algorithm. It is suitable for low and moderate Mach number flows where transient behavior is important.

rhoCentralFoam

`rhoCentralFoam` is a transient, density-based solver designed for compressible flows. It employs a central-upwind scheme suitable for high-speed flows, including transonic, supersonic, and hypersonic regimes. The solver effectively captures shock waves and discontinuities using a robust numerical method based on the Kurganov and Tadmor central scheme.
`rhoCentralFoam` employs a central-upwind scheme with the following features:

- Kurganov-Tadmor central scheme for shock capturing.
- Flux splitting into left-going and right-going components.
- MUSCL (Monotonic Upstream-Centered Scheme) for second-order spatial accuracy.
- Runge-Kutta time integration.
- Harten-Lax-van Leer (HLL)-type numerical flux formulation.

These features enhance stability and accuracy in capturing shocks, rarefactions, and contact discontinuities effectively. While it has some limitations, particularly for steady-state problems, it remains a robust, efficient, and widely used solver in the computational fluid dynamics (CFD) community.

sonicFoam

`sonicFoam` is a transient, pressure-based solver for compressible flows that supports subsonic, transonic, and supersonic conditions. It uses the PISO algorithm for time marching and pressure-velocity coupling.

dnsFoam

The dnsFoam solver in OpenFOAM stands for Direct Numerical Simulation Foam. It's a computational fluid dynamics (CFD) solver that allows for the direct simulation of turbulent flows without using turbulence models. Instead of modeling turbulence with empirical relations, dnsFoam resolves all turbulent scales of motion directly by solving the Navier-Stokes equations numerically on a grid that resolves all length scales of turbulence. The dnsFoam solver in OpenFOAM

is a transient (time-dependent), pressure-velocity coupled solver that typically uses the PISO (Pressure-Implicit with Splitting of Operators) algorithm for time-accurate simulations.
The following table presents a comparative summary of the solvers described above, highlighting the flow regime, numerical approach, and their capability to handle shocks and turbulence.

Solver	Flow Regime	Approach	Shock Handling	Turbulence Support
`simpleFoam`	Incompressible, steady-state	Pressure-based	Low	RANS
`pimpleFoam`	Incompressible, transient	Pressure-based	Low	RANS, LES
`pisoFoam`	Incompressible, transient	Pressure-based	Low	RANS, LES
`rhoSimpleFoam`	Compressible, steady-state	Pressure-based	Moderate	RANS
`rhoPimpleFoam`	Compressible, transient	Pressure-based	Moderate	RANS, LES
`rhoPisoFoam`	Compressible, transient	Pressure-based	Moderate	RANS, LES
`rhoCentralFoam`	Compressible, high-speed	Density-based	Good	RANS, LES
`sonicFoam`	Compressible, subsonic/supersonic	Pressure-based	Moderate	RANS, LES
`dnsFoam`	Incompressible, transient	Pressure-based	Low	DNS

Table 2.1. Comparison of OpenFOAM Solvers for incompressible and compressible flows

2.6 Turbulence Modeling

Turbulence modeling is a cornerstone of computational fluid dynamics (CFD), essential for predicting complex flow behavior in engineering and scientific applications. The Reynolds-Averaged Navier-Stokes (RANS) approach, widely used for its efficiency, simplifies turbulence by averaging its effects over time, making it practical for many industrial problems despite its limitations in capturing finer flow structures. Large Eddy Simulation (LES) takes a step further, directly resolving larger turbulent eddies while modeling only the smaller, more universal scales, offering a better balance between accuracy and computational cost. At the most detailed level, Direct Numerical Simulation (DNS) resolves all turbulent scales without approximation, providing the highest fidelity at an extraordinary computational expense. Each of these methods—RANS, LES, and DNS—comes with its own trade-offs, shaping their use in different contexts where turbulence must be understood and controlled.

2.6.1 Reynolds-Averaged Navier-Stokes (RANS)

The RANS approach is one of the most common methods used in turbulence modeling. It implies averaging the Navier-Stokes equations over time, which effectively smooths out fluctuations in turbulence to provide a steady-state solution. This method greatly simplifies computational requirements and is particularly useful for applications where the flow is stable or slightly unsteady. RANS models are widely used in industrial applications because of their robustness and low computational costs. However, they may be less accurate in predicting complex flows with significant separation or strong instability. The key to obtain RANS equations is to use the Reynolds decomposition which divide the velocity field into a mean component (U) and a fluctuating component (u'):

$$u = U + u' \tag{2.70}$$

The Reynolds-averaged Navier Stokes (RANS) equation in tensor form, derived from the incompressible form of the Navier Stokes equations, can be written as:

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial(\rho U_i U_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right] \tag{2.71}$$

where μ is the molecular viscosity. The Reynolds decomposition introduces a term called Reynolds stress ($-\rho\overline{u_i' u_j'}$) when applied to the Navier-Stokes equations, that can be modeled with the Boussinesq hypothesis [2] as follows:

$$-\rho\overline{u_i' u_j'} = \mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3}\frac{\partial U_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k\delta_{ij} \tag{2.72}$$

where μ_t is the turbulent eddy viscosity and δ_{ij} is the Kronecker Delta:

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.73}$$

The main challenge with Reynolds-Averaged Navier-Stokes (RANS)

equations lies in the imbalance between the number of unknowns and equations. In RANS modeling, the equations describing the mean flow properties are insufficient to fully determine all the required variables. This inadequacy arises because the equations for the first moments (mean flow quantities) depend on correlations that involve pairs of variables (second moments). To accurately represent these second moments, RANS equations incorporate terms involving triple correlations. These correlations introduce additional complexity. This problem remains at all correlation levels and, in turbulence, is known as the closure problem. Currently, there are three approaches to the closure problem. The first is RANS closure modeling, which involves ending the number of equations at a finite value and then closing the resulting system using approximate models. The second, known as Large-Eddy Simulation (LES), use modeling the fields near the wall of the body using RANS and a direct numerical simulation of large-scale turbulent fluctuations. The third, known as Direct Numerical Simulations (DNS) involves fully numerically solving the time-dependent equations of motion and then Reynolds averaging the computational output to determine mean-flow quantities.

2.6.2 RANS closure models

The closure problem has driven the development of approximate models to complete the RANS system. Due to their critical role in practical applications—such as weather forecasting—RANS closure models have been an area of active research for over a century and remain a central focus in turbulence modeling today. This section presents a truncated overview of the essential elements leading to the most known turbulence models that can be divided into two categories: the one equation models like the Spalart-Allmaras model, and the two-equation models like the $k - \varepsilon$ model and the $k - \omega$ model.

Spalart-Allmaras model

The Spalart-Allmaras (SA) model is a one-equation turbulence model designed primarily for aerodynamic flows, particularly those involving wall-bounded and external flows around airfoils, wings, and vehicles. This model introduces a transport equation for a modified **eddy viscosity** ($\tilde{\nu}$), making it computationally efficient while still capturing important turbulence effects.
Hereafter, we report the Standard Spalart-Allmaras Model [41].

The turbulent **eddy viscosity** is given by:

$$\nu_t = \tilde{\nu} f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi := \frac{\tilde{\nu}}{\nu} \tag{2.74}$$

and the transport equation for the modified eddy viscosity is:

$$\begin{aligned} \frac{\partial \tilde{\nu}}{\partial t} + u_j \frac{\partial \tilde{\nu}}{\partial x_j} &= C_{b1}[1 - f_{t2}]\tilde{S}\tilde{\nu} + \\ + \frac{1}{\sigma}\left\{\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + C_{b2}|\nabla\tilde{\nu}|^2\right\} &+ \\ - \left[C_{w1} f_w - \frac{C_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 &+ f_{t1}\Delta U^2 \end{aligned} \tag{2.75}$$

where d is the distance from the field point to the nearest wall and

$$\tilde{S} \equiv S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \quad S = \sqrt{2 S_{ij} S_{ij}}. \tag{2.76}$$

Additional definitions are given by the following equations:

$$C_{w1} = \frac{C_{b1}}{\kappa^2} + \frac{(1 + C_{b2})}{\sigma} \tag{2.77}$$

$$f_w(r) = g\left[\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\right]^{1/6}, \quad g = r + C_{w2}(r^6 - r), \quad r \equiv \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \tag{2.78}$$

$$f_{t1} = C_{t1} g_t \exp\left(-C_{t2}\frac{\omega_t^2}{\Delta U^2}[d^2 + g_t^2 d_t^2]\right) \quad f_{t2} = C_{t3}\exp(-C_{t4}\chi^2) \tag{2.79}$$

The rotation tensor is given by

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right), \tag{2.80}$$

and ΔU^2 is the norm of the difference between the velocity at the trip (usually zero) and that at the field point we are considering. All the constant are written in table 2.2:

Constant	Value
σ	2/3
C_{b1}	0.1355
C_{b2}	0.622
κ	0.41
C_{w2}	0.3
C_{w3}	2
C_{v1}	7.1
C_{t1}	1
C_{t2}	2
C_{t3}	1.1
C_{t4}	2

Table 2.2. Spalart-Allmaras constants

There are several approaches to adapting the model for compressible flows. In all cases, the turbulent dynamic viscosity is computed from

$$\mu_t = \rho\tilde{\nu} f_{v1} \tag{2.81}$$

where ρ is the local density.

$k-\varepsilon$ model

This two-equations model solves transport equations for turbulent kinetic energy (k) and dissipation rate (ε):

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k u_i}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k + P_b - \rho\varepsilon + S_k \tag{2.82}$$

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon u_i}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}(P_k + C_{3\varepsilon}P_b) - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} + S_\varepsilon \tag{2.83}$$

Where u_i represents the component of the velocity in the correspond-

ing direction, E_{ij} represents the component of the strain tensor, μ_t represents the turbulent viscosity and has the following equation:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{2.84}$$

Moreover, P_k is the production of turbulent kinetic energy term due to mean velocity shear, P_b is the production of TKE due to buoyancy, S_k and S_ε are source terms defined by the user. The equation contains some calibration constants σ_k, σ_ε, $C_{1\varepsilon}$ and $C_{2\varepsilon}$. The value of these constants are derived from experimental results, and the standard values used are: $C_\mu = 0.09$, $\sigma_k = 1.00$, $\sigma_\varepsilon = 1.30$, $C_{1\varepsilon} = 1.44$ and $C_{2\varepsilon} = 1.92$.

$k - \omega$ model

The $k-\omega$ turbulence model is widely used for simulating turbulent flow conditions within the framework of Reynolds-averaged Navier-Stokes (RANS) models. It falls under the category of two-equation models, which augment the conservation equations with two additional transport equations (PDEs). These equations account for the temporal and spatial variations of turbulent energy (k) and specific turbulent dissipation rate (ω). k represents the turbulent kinetic energy, indicating the energy content of turbulence, while ω characterizes the rate at which turbulence dissipates per unit of k. This dissipation rate is also known as the turbulence scale. The transport equation for turbulent kinetic energy and for the turbulent dissipation rate are given in equation 2.85 and 2.86 as described in Wilcox [46] paper:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j k) = \rho \tau_{ij} \frac{\partial u_i}{\partial x_j} - \rho\varepsilon + \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma^* \frac{\rho k}{\omega}\right)\frac{\partial k}{\partial x_j}\right] \tag{2.85}$$

$$\begin{aligned}\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j \omega) = \alpha \frac{\omega}{k} \rho \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta \rho \omega^2 + \sigma_d \frac{\rho}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} + \\ + \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma \frac{\rho k}{\omega}\right)\frac{\partial \omega}{\partial x_j}\right]\end{aligned} \tag{2.86}$$

where u_i represents the component of the velocity in the corresponding direction, τ_{ij} is the shear stress tensor, μ is the dynamic viscosity,

$\nu_t = k/\omega$ is the kinematic eddy viscosity, $\varepsilon = \beta^*\omega k$, $\alpha = 0.52$, $\beta^* = 0.09$ and $\sigma^* = 0.60$, $\sigma = 0.50$ and

$$\sigma_d = \begin{cases} 0, & \frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \leq 0 \\ \sigma_{d0}, & \frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} > 0 \end{cases}, \qquad \sigma_{d0} = 0.125 \tag{2.87}$$

Specifically, the model is as accurate for attached boundary layers, backward-facing steps, and mildly separated incompressible flows. Also, $k - \omega$ model predicts reasonably close agreement with measured properties of shock-separated flows for transonic, supersonic, and hypersonic regimes.

2.6.3 Unsteady RANS (URANS)

URANS extends the RANS approach by allowing time-dependent variations in the flow field, making it capable of capturing unsteady phenomena. It still uses Reynolds averaging of the Navier-Stokes equations but does not calculate the time-averaged flow as rigorously as RANS. This means that URANS can model large-scale transient flow characteristics and oscillatory behaviors that are typical in many practical engineering systems, such as vortex detachment from building corners. Although URANS improves RANS in terms of capturing instability, it still uses eddy viscosity models that may not adequately resolve finer turbulent structures. Additionally, while URANS requires more computational resources than RANS, it remains significantly more efficient than higher-fidelity turbulence models like Large Eddy Simulation (LES) or Delayed Detached Eddy Simulation (DDES). This balance between accuracy and efficiency makes URANS a practical choice for many engineering applications where large-scale unsteady behavior plays a critical role. However, URANS has limitations, particularly in handling complex turbulent flows. Due to its reliance on eddy-viscosity models, it may struggle to accurately predict fine turbulent structures. Moreover, despite being time-dependent, URANS still averages the flow over time, potentially smoothing out important transient details. Another drawback is its reduced effectiveness in strongly separated flows, as it lacks the refinement of LES or hybrid methods in capturing separation and reattachment phenomena. Despite these limitations, URANS remains a widely used approach in engineering applications requiring a compromise between accuracy and computational efficiency.

2.6.4 Large Eddy Simulation (LES)

Large Eddy Simulation (LES) is a mathematical model used in CFD to study turbulent phenomena. Initially proposed in 1963 by Joseph Smagorinsky [38] for modeling atmospheric currents, it is still the subject of great interest both because of its possible developments and because it stands as a middle ground between RANS-type modeling (faster but more approximate) and DNS direct numerical simulation (more exact but computationally very expensive). LES basic idea is to reduce computational cost by ignoring the smallest length scales, which are the most expensive to resolve, through low-pass filtering of the Navier–Stokes equations. This filtering can be seen as a combination of time- and spatial-averaging, effectively removing small-scale information from the numerical solution. However, this information is not irrelevant, because the effects of the small scales on the flow field must still be modeled. For instance, a generic model of filtering is the following

$$\overline{u(x,t)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(\bar{x},\bar{t})F(x-\bar{x},t-\bar{t})\mathrm{d}t\mathrm{d}\bar{x}, \tag{2.88}$$

where F is a convolution function different for any type of filter. Using the above filter definition, the field u may be split up into a filtered and sub-filtered (denoted with a prime) portion, as

$$u = \bar{u} + u'. \tag{2.89}$$

It is important to note that the large eddy simulation filtering operation does not satisfy the properties of a Reynolds operator. The governing equations of LES are obtained by filtering the Navier-Stokes equations both for incompressible flows and compressible flows. The former approach generate the filtered advection term that can be written as

$$\overline{u_i u_j} = \tau_{ij} + \bar{u}_i\bar{u}_j \tag{2.90}$$

where τ_{ij} is the residual stress tensor, so that the filtered incompressible momentum balance equation becomes:

$$\frac{\partial\bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\bar{u}_i\bar{u}_j) = -\frac{1}{\rho}\frac{\partial\bar{p}}{\partial x_i} + 2\nu\frac{\partial\bar{S_{ij}}}{\partial x_j} - \frac{\partial\tau_{ij}}{\partial x_j} \tag{2.91}$$

where $\bar{S}_{ij}$ is the filtered strain rate tensor:

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.92}$$

The Smagorinsky model treats dissipation of kinetic energy at sub-grid scales as analogous to molecular diffusion and so, it relates the filtered strain rate tensor to the stress tensor τ_{ij}^r:

$$\tau_{ij}^r = \tau_{ij} - \frac{1}{3}\delta_{ij}\tau_{kk} = -2\nu_t \bar{S}_{ij} \tag{2.93}$$

where ν_t is the turbulent eddy viscosity. The first SubGrid Scale (SGS) model developed by Smagorinsky and used in the first LES, models the eddy viscosity as follows

$$\nu_t = C_s \Delta^2 |\bar{S}| \tag{2.94}$$

where Δ is the grid size and C_s is the Smagorinsky constant.
For the governing equations of compressible flow, each equation is filtered. Favre [11] proposed a density-weighted filtering operation, called Favre filtering, defined for an arbitrary quantity ϕ as:

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\overline{\rho}} \tag{2.95}$$

The momentum balance equation of the Navier-Stokes equations for compressible flows becomes:

$$\frac{\partial \overline{\rho}\tilde{u_i}}{\partial t} + \frac{\partial \overline{\rho}\tilde{u_i}\tilde{u_j}}{\partial x_j} + \frac{\partial \overline{p}}{\partial x_i} - \frac{\partial \tilde{\sigma}_{ij}}{\partial x_j} = -\frac{\partial \overline{\rho}\tau_{ij}^r}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\overline{\sigma}_{ij} - \tilde{\sigma}_{ij}\right) \tag{2.96}$$

where σ_{ij} is the shear stress tensor, given for a Newtonian fluid by:

$$\sigma_{ij} = 2\mu(T)S_{ij} - \frac{2}{3}\mu(T)\delta_{ij}S_{kk} \tag{2.97}$$

and the last term of the right hand side of the equation 2.96 represents a sub-filter viscous contribution from evaluating the viscosity $\mu(T)$ using the Favre-filtered temperature $\tilde{T}$. The subgrid stress tensor for the Favre-filtered momentum field is given by

$$\tau_{ij}^r = \widetilde{u_i \cdot u_j} - \tilde{u_i}\tilde{u_j} \tag{2.98}$$

2.6.5 Detached Eddy Simulation (DES)

DES was born from the necessity to resolve, with reduced computational cost, high–Reynolds number, massively separated flows, which must be addressed in many fields such as aerospace and ground transportation, as well as in atmospheric forecast. DES was initially proposed by Spalart [40] in 1997 and it combines RANS and LES approaches. The basic formulas of DES was based on the substitution in the Spalart–Allmaras turbulence model of the wall distance d with a new wall distance $\tilde{d}$ that controls the eddy viscosity:

$$\tilde{d} = \min(d, C_{DES}\Delta) \tag{2.99}$$

where C_{DES} is a model constant of order 1 and $\Delta = max(\Delta x, \Delta y, \Delta z)$.

2.6.6 Direct Numerical Simulation (DNS)

DNS means Direct Numerical Simulation and it resolves all scales down to the Kolmogorov scales:

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4} \quad \text{(length scale)} \tag{2.100}$$

$$\tau_\eta = \left(\frac{\nu}{\varepsilon}\right)^{1/2} \quad \text{(time scale)} \tag{2.101}$$

In fact, DNS generates complete three-dimensional and time-dependent solution of the Navier–Stokes equations to obtain results for the instantaneous fluid velocity as a function of position and time. Unfortunately this is not possible for a the majority of physical flows. It is true that Direct numerical simulation has been applied to a limited number of geometrically simple flow situations, such as flow in a channel or over a flat plate. The computational requirements for DNS scale with Reynolds number is:

$$N \sim Re^3 \tag{2.102}$$

where N is the total number of grid points required. In practical turbulent flow simulations, such as those with a Reynolds number around 10^6, the number of grid points required would be at least

$N \sim 10^{18}$. This presents a formidable computational and it would not be feasible given the current technology.

Chapter 3

A New Density-Based Solver

This solver, proposed by Modesti and Pirozzoli [26] in 2017, addresses the numerical challenges of simulating fluid dynamics in situations involving significant density changes, complex energy transfer mechanisms and temperature variations. By employing advanced numerical discretization techniques and solving coupled energy and momentum conservation equations, rhoEnergyFoam provides researchers and engineers with a powerful computational framework for investigating thermodynamic phenomena across diverse applications, from aeronautical engineering to industrial thermal management systems. The following methodology section will delve into the mathematical formulation, numerical algorithms, and computational strategies that underpin the rhoEnergyFoam solver, elucidating the precise numerical methods used to transform fundamental conservation laws into computationally tractable algorithms capable of accurately predicting fluid behavior under complex thermodynamic conditions.

3.1 rhoEnergyFoam

rhoEnergyFoam is a density-based solver developed in the OpenFoam library. It exploits the predictive capabilities of low-diffusion numerical schemes useful for the applied study of flows around complex geometries. The spatial discretization of convective terms in the Navier-Stokes equations is based on a robust energy-preserving nu-

merical flow, and the numerical diffusion inherited from the AUSM scheme is added limitedly to the vicinity of shock waves, or wherever spurious numerical oscillations are detected. The solver is capable of preserving the total kinetic energy in the inviscid limit, and bears significantly less numerical diffusion than typical industrial solvers, with greater predictive power, as demonstrated through a series of test cases including URANS, LES and DNS of turbulent flows.
Let's consider the Navier-Stokes equations for a compressible ideal gas, integrated over an arbitrary volume V

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{u}\,\mathrm{d}V + \sum_{i=1}^{3}\int_{\partial V}(\mathbf{f}_i - \mathbf{f}_i^v)\cdot n_i\,\mathrm{d}S = 0 \tag{3.1}$$

where n_i is the outward normal, and

$$\mathbf{u} = \begin{Bmatrix}\rho\\ \rho u_i\\ \rho E\end{Bmatrix}, \quad \mathbf{f}_i = \begin{Bmatrix}\rho u_i\\ \rho u_i u_j + p\delta_{ij}\\ \rho u_i H\end{Bmatrix}, \quad \mathbf{f}_i^v = \begin{Bmatrix}0\\ \sigma_{ij}\\ \sigma_{ik}u_k - q_i\end{Bmatrix}, \tag{3.2}$$

are the vector of conservative variables, and the associated Eulerian and viscous fluxes, respectively. Here ρ is the density, u_i is the velocity component in the $i-th$ coordinate direction, p is the thermodynamic pressure, $E = e+u^2/2$ is the total energy per unit mass, $e = RT/(\gamma-1)$ is the internal energy per unit mass, $H = E+p/\rho$ is the total enthalpy, R is the gas constant, $\gamma = c_p/c_v$ is the specific heat ratio, σ_{ij} is the viscous stress tensor, and q_i is the heat flux vector.
The boundary Eulerian flux in equation 3.1 is approximated on a polyhedral cell O (see figure 3.1 for illustration) as follows

$$\frac{1}{V}\int_{\partial V}\mathbf{f_i}n_i\mathrm{d}S \approx \sum_N(\mathbf{f_i}n_i)_{ON}\Delta S_{ON} = \sum_N \hat{\mathbf{f}}_{ON}\Delta S_{ON}. \tag{3.3}$$

Let $\hat{\mathbf{f}}_{ON}$ represent the numerical flux at the interface shared between a given cell and its neighboring cell N. The interface area is denoted by ΔS_{ON}, and $\sum_N$ signifies summation over all cell faces.
Following the standard AUSM methodology [24], we decompose the Euler fluxes in (3.2) into two components: one representing convection and the other accounting for pressure, given as:

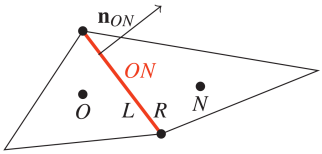


Figure 3.1. Computational cell for numerical flux evaluation. ON represents the interface between the owner cell O and the neighboring cell N, with n_{ON} as the outward normal for O. L and R denote the limit states on either side of the interface, where $L \equiv O$ and $R \equiv O$ in the case of first-order reconstruction.

$$\mathbf{f}_i = f_i + p_i = \begin{Bmatrix} \rho u_i \\ \rho u_i u_j \\ \rho u_i H \end{Bmatrix} + \begin{Bmatrix} 0 \\ p\delta_{ij} \\ 0 \end{Bmatrix}. \tag{3.4}$$

The corresponding numerical fluxes can be expressed as the sum of a central contribution and a diffusive contribution,both for the convective terms:

$$\hat{f}_{ON} = \hat{f}^c_{ON} + \hat{f}^D_{ON} \tag{3.5}$$

and the pressure terms:

$$\hat{p}_{ON} = \hat{p}^C_{ON} + \hat{p}^D_{ON}. \tag{3.6}$$

The central part of the convective flux is determined using the following relation [31]:

$$\hat{f}^c_{ON} = \frac{1}{8}(\rho_0 + \rho_N)(u_{n0} + u_{nN})(\boldsymbol{\varphi}_0 + \boldsymbol{\varphi}_N), \tag{3.7}$$

where $\boldsymbol{\varphi} = (1, u_i, H)^T$, and the pressure flux is obtained via a standard central interpolation approach:

$$\hat{p}^C_{ON} = \frac{1}{2}(p_0 + p_N). \tag{3.8}$$

Unlike conventional central differencing schemes, the numerical flux in equation (3.7) ensures discrete conservation of the total kinetic energy

transported by convection, albeit with significant nonlinear stability properties. In fact, this central numerical flux has been observed to maintain stability in fully resolved direct numerical simulations (DNS) on Cartesian grids or slightly distorted meshes [31, 32]. Nevertheless, when applied to practical computational fluid dynamics problems using unstructured meshes, especially in the presence of shock waves, it becomes necessary to introduce a certain level of numerical diffusion. Consequently, the diffusive fluxes in equation should be selectively activated in regions where resolution deteriorates. To assess local smoothness and determine where numerical diffusion should be applied, a classical shock sensor is employed [9].

$$\theta = max\left(\frac{-\nabla\cdot\mathbf{u}}{\sqrt{(\nabla\cdot\mathbf{u})^2+(\nabla\times\mathbf{u})^2+u_0^2/L_0}},0\right) \in [0,1], \tag{3.9}$$

Where u_0 and L_0 represent appropriate velocity and length scales [33], they are defined such that $\theta \approx 0$ in smooth regions and $\theta \approx 1$ in the presence of shocks.
For smooth flows (i.e., without shocks), we have observed that numerical stability can be improved with minimal accuracy loss by applying the artificial diffusion term exclusively to the pressure flux, in an amount proportional to θ_{ON}.

$$\theta_{\text{ON}} = \frac{1}{2}(\theta_O + \theta_N) \tag{3.10}$$

To accurately capture shock waves, it is necessary to also activate the convective diffusive flux whenever θ_{ON} exceeds a specified threshold (θ^*, set to 0.05 unless stated otherwise). Thus, the diffusive numerical fluxes used in equations (3.5, 3.6) can be concisely expressed as follows:

$$\hat{f}_{\text{ON}} = (\text{IC})H(\theta_{\text{ON}}-\theta^*)\tilde{p}_{\text{ON}}^{AUSM}, \quad \tilde{p}_D = \text{IP}\theta_{\text{ON}}\tilde{p}_D^{AUSM}, \tag{3.11}$$

(IC) and (IP) are flags that regulate the activation of convective and pressure diffusive fluxes, respectively. H represents the Heaviside step function, while the artificial diffusion fluxes are derived from the AUSM scheme, as outlined in section 2.4.3. Recommended values for (IC) and (IP), based on the type of numerical simulation being performed, are provided in table 3.1.

Mode	Intent	IC	IP
A	Fully resolved smooth flows	0	0
B	Unresolved smooth flows	0	1
C	Shocked flows	1	1

Table 3.1. Operational modes of `rhoEnergyFoam`, with corresponding suggested values for the flags in equation 3.11.

The discretization of viscous fluxes is based on standard second-order approximations for unstructured meshes [17] and is implemented using the `fvc::laplacian()` primitive in OpenFOAM.
The resulting semi-discretized system of ordinary differential equations, say

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = R(\mathbf{u}), \tag{3.12}$$

is advanced in time using a low-storage third-order, three-stage Runge–Kutta algorithm, where the substeps are the following:

$$\begin{cases} \mathbf{u}' = \mathbf{u}^n + \gamma_1 \Delta t \mathbf{R}(\mathbf{u}^n) + \rho_1 \Delta t \mathbf{R}(\mathbf{u}^{n-1}) \\ \mathbf{u}'' = \mathbf{u}' + \gamma_2 \Delta t \mathbf{R}(\mathbf{u}') + \rho_2 \Delta t \mathbf{R}(\mathbf{u}^n) \\ \mathbf{u}''' = \mathbf{u}'' + \gamma_3 \Delta t \mathbf{R}(\mathbf{u}'') + \rho_3 \Delta t \mathbf{R}(\mathbf{u}') \end{cases} \tag{3.13}$$

where the constants are displayed in table 3.2.
Referring to figure 2.3, the AUSM convective and pressure flux to be used in equation (3.11) are given below, based on the AUSM+ -up formulation [23]:

$$\hat{\mathbf{f}}_{ON}^{AUSM} = -\frac{C_{ON}}{2}\left[\left(\frac{1}{2}\delta m_{ON} - |M_{ON}|\right)\varphi_L + \left(\frac{1}{2}\delta m_{ON} + |M_{ON}|\right)\varphi_R\right]. \tag{3.14}$$

$$\hat{\mathbf{p}}_{ON}^{AUSM} = -\frac{1}{2}\delta \mathbf{p}_{ON}, \tag{3.15}$$

$$M_{ON} = \frac{M_R + M_L}{2} - \frac{1}{2}\delta m_{ON}, \tag{3.16}$$

Constant	Value
γ_1	$\frac{8}{15}$
γ_2	$\frac{5}{12}$
γ_3	$\frac{3}{4}$
ρ_1	0
ρ_2	$-\frac{17}{60}$
ρ_3	$-\frac{5}{12}$

Table 3.2. Coefficient of the low-storage third-order, three-stage Runge–Kutta algorithm

$$\delta m_{ON} = [\Delta\mathcal{M}(M_R) - \Delta\mathcal{M}(M_L)] - 2M_p, \tag{3.17}$$

$$\Delta\mathcal{M}(M) = \mathcal{M}^{+}_{(4)}(M) - \mathcal{M}^{-}_{(4)}(M), \tag{3.18}$$

$$M_p = \frac{k_p}{f_a}\max\left(1 - \sigma\overline{M}^2, 0\right)\frac{2(p_R - p_L)}{(\rho_L + \rho_R)c^2_{ON}}. \tag{3.19}$$

The speed of sound at the cell interface is evaluated as

$$c_{ON} = \frac{(c_L + c_R)}{2}, \quad \text{and} \quad \overline{M}^2 = \frac{(u^2_{nL} + u^2_{nR})}{(2c^2_{ON})}. \tag{3.20}$$

$$M_0^2 = \min(1, \max(\overline{M}^2, M^2_\infty)), \tag{3.21}$$

$$f_a(M_0) = M_0(2 - M_0), \tag{3.22}$$

with $k_p = 0.25$, $k_u = 0.75$, and $\sigma = 1$. The diffusive-like pressure flux is given by

$$\delta p_{ON} = [p_R\Delta\mathcal{P}(M_R) - p_L\Delta\mathcal{P}(M_L)] - 2M_u. \tag{3.23}$$

$$\Delta\mathcal{P}(M) = \mathcal{P}^{+}_{(5)}(M) - \mathcal{P}^{-}_{(5)}(M), \tag{3.24}$$

where

$$M_u = -\frac{1}{2}k_u \mathcal{P}^+_{(5)}(M_L)\mathcal{P}^-_{(5)}(M_R)(\rho_L+\rho_R)(f_a c_{ON})(u_{nR}-u_{nL}). \quad (3.25)$$

The subscript L, R refers to the two sides of the cell interface, which have been reconstructed through the Minmod limiter, also available in the OpenFOAM library. We further define the split Mach numbers $\mathcal{M}_{(m)}$ as m-th degree polynomials

$$\mathcal{M}^{\pm}_{(1)}(M) = \frac{1}{2}(M \pm |M|), \quad (3.26)$$

$$\mathcal{M}^{\pm}_{(2)}(M) = \pm\frac{1}{4}(M \pm 1)^2, \quad (3.27)$$

$$\mathcal{M}^{\pm}_{(4)}(M) = \begin{cases} \mathcal{M}^{\pm}_{(1)}(M), & \text{if } |M| \geq 1, \\ \mathcal{M}^{\pm}_{(2)}(M)(1 \mp 16\beta\mathcal{M}^{\mp}_{(2)}(M)), & \text{if } |M| < 1. \end{cases} \quad (3.28)$$

$P^+_{(5)}$ is also defined in terms of the split Mach numbers, as follows:

$$P^+_{(5)}(M) = \begin{cases} \frac{1}{M}\mathcal{M}^{+}_{(1)}(M), & \text{if } |M| \geq 1, \\ \mathcal{M}^{+}_{(2)}(M)[(2 \pm M) \mp 16\alpha M\mathcal{M}^{\mp}_{(2)}(M)], & \text{if } |M| < 1. \end{cases} \quad (3.29)$$

Following Liou [23], we set $\alpha = 3/16(-4 + 5f_a^2)$, $\beta = 1/8$.
From its original version, working in OpenFOAM's version v2006, `rhoEnergyFoam` was then optimized in terms of computational cost and accuracy, and was ported to the ESI OpenFOAM v2412 version. Moreover, the Adaptive Mesh Refinement (AMR) already present in OpenFOAM, has been added as a new feature in `rhoEnergyFoam` and its results can be seen in section 4.3.

3.1.1 Adaptive Mesh Refinement (AMR)

Adaptive Mesh Refinement (AMR) is a technique that dynamically refines the computational mesh based on predefined criteria and specific

control parameters, such as refinement and unrefinement levels, as well as expansion ratios. In OpenFOAM, AMR is supported by solvers that incorporate dynamic mesh capabilities, with compatibility restricted to hexahedral meshes: itis available in solvers from the PIMPLE family, and now available also in `rhoEnergyFoam`. Employing an accurate and stable numerical method is crucial when utilizing AMR. If an unbounded or excessively diffusive method (both in space and time) is used, the refinement strategy may fail to effectively capture the quantity of interest, potentially leading to overshoots and undershoots in the solution. Within OpenFOAM, all dynamic mesh capabilities are controlled in the dictionary `dynamicMeshDict`, which is located in the `constant` directory. To appropriately control the time step as the mesh is refined, the recommended approach in OpenFOAM is to enable the `adjustableTimeStep` option and, maintaining a Courant-Friedrichs-Lewy (CFL) number below 1, ideally around 0.5, helps to increase the accuracy of the results. It is important to note that AMR introduces a significant computational overhead, which should be considered when designing simulations. AMR is widely used in various engineering and scientific fields due to its ability to enhance simulation efficiency. In fluid dynamics, it plays a crucial role in turbulent flow simulations, shock capturing, and aeroacoustics by refining areas with high vorticity, shear layers, or boundary layers. In weather and climate modeling, AMR is employed to track the evolution of hurricanes and storms by refining mesh in regions of high vorticity or pressure gradients, thereby improving large-scale atmospheric simulations. It is used in combustion and reactive flow simulations to refine mesh around flame fronts and regions with steep temperature and species concentration gradients. It is also extensively used in structural mechanics and materials science, where it helps to refine regions where cracks propagate in materials and enhances stress analysis by providing higher resolution in high-stress zones. By refining only the necessary regions, AMR allows engineers and scientists to conduct high-fidelity simulations without excessive computational costs, making it a powerful tool for numerical simulations across different disciplines.

Chapter 4

Test Cases and Results

In this chapter, we present a series of test cases that show the capability of rhoEnergyFoam evaluating different test cases where all the three operational modes outlined in table 3.1 are analyzed, aiming to evaluate the energy-preserving properties of the proposed solver in comparison to standard OpenFOAM solvers. To assess numerical diffusion, we analyze Euler turbulence and Taylor–Green flow. To assess the capability of AMR, we tested a subsonic turbulent flow past a circular cylinder. To examine the effectiveness of background numerical diffusion in smooth flows, we conduct Unsteady Reynolds-Averaged Navier–Stokes (URANS) and Detached Eddy Simulations (DES) of subsonic turbulent flow past a circular cylinder. To test the accuracy of the solver we simulate the Sod's shock tube. Finally, the shock-capturing capabilities of rhoEnergyFoam are tested using three classical flow cases: inviscid supersonic flow past a forward-facing step, transonic flow past an RAE airfoil, and transonic flow past the ONERA M6 wing.

4.1 Decaying Isotropic Turbulence

The study of a decaying isotropic turbulence case in the literature serves as a fundamental test for understanding turbulence dynamics in the absence of external forcing or mean flow effects. Isotropic turbulence, characterized by uniform statistical properties in all directions, provides a simplified yet insightful framework to investigate how turbulence evolves over time. Researchers use this case to examine the

decay laws of turbulent kinetic energy, the role of vortex structures, and the mechanisms governing the dissipation of energy. It also serves as a benchmark for testing turbulence models, numerical schemes, and theoretical predictions. Since isotropic turbulence eliminates directional biases, it allows for a more general study of turbulence physics that can be applied to a wide range of engineering and geophysical flows. Also, the decay process sheds light on turbulence behavior, understanding mixing, diffusion, and energy cascade processes in both natural and industrial settings. In this study, the decay of isotropic turbulence is considered to quantify the energy conservation properties of rhoEnergyFoam, resolving Navier-Stokes equations in the absence of kinematic viscosity ($\nu = 0$). This means that the NS equations becomes the Euler equations as follows

$$\begin{cases} \nabla \cdot u = 0 \\ \frac{\partial u}{\partial t} + u \cdot \nabla u = -\frac{1}{\rho_0} \nabla p \end{cases} \tag{4.1}$$

Numerical simulation was conducted on a structured Cartesian grid consisting of 32^3 cells (32 cells per side), with uniform spacing and constant timestep, imposing a Courant number (CFL) of 1. Random initial conditions are used with prescribed energy spectrum

$$E(k) = 16\sqrt{\frac{2}{\pi}}\frac{u_0^2}{k_0}\left(\frac{k}{k_0}\right)^2 e^{-2(k/k_0)^2} \tag{4.2}$$

where $k_0 = 4$ is the most energetic mode, and u_0 is the initial r.m.s. velocity. The Mach number adopted for the simulation was $Ma = u_0/c_0 = 0.01$, where $c_0 = \sqrt{\gamma}$ is the adimensionalized speed of sound. Time is made nondimensional with respect to the eddy turnover time $\tau = 2/k_0 u_0$ The analysis was carried out by comparing the results obtained with rhoEnergyFoam in three different modes (A, B and C) and rhoCentralFOAM. In addition, the case was also evaluated using the dnsFOAM solver for a more in-depth comparison.

$$\mathrm{CFL} = \max\left(\frac{(u_0 + c_0)\Delta t}{\Delta x}\right). \tag{4.3}$$

The evolution of turbulence kinetic energy defined as:

$$K = \frac{1}{2}\sum_i (u_k u_k)_i V_i, \tag{4.4}$$

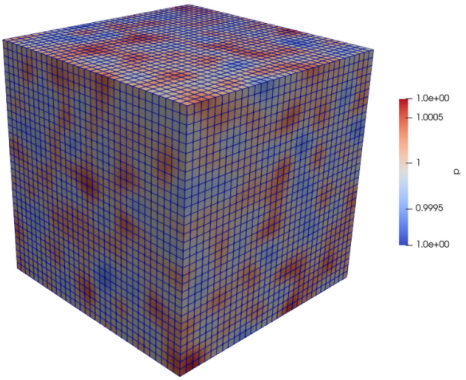


Figure 4.1. Structured mesh for numerical simulation of decaying isotropic turbulence.

where u_k represents the velocity of the cell i, V_i is the volume of cell, can be seen in figure 4.2. From an aerospace engineering perspective,

rhoEnergyFoam - mode A	
rhoEnergyFoam - mode B	
rhoEnergyFoam - mode C	
rhoCentralFoam	
dnsFoam	

accurately capturing turbulent kinetic energy dynamics is crucial, particularly in high-speed aerodynamic flows where shock-boundary layer interactions and numerical diffusion can significantly impact predictive accuracy. The numerical study is conducted with convective diffusive fluxes initially deactivated (mode A) to assess the upper bound of numerical diffusion in shock-capturing simulations.
To benchmark the solvers, results from `rhoCentralFoam` and the incompressible Direct Numerical Simulation (DNS) solver `dnsFoam` from OpenFOAM are also presented. The findings indicate that neither of the standard OpenFOAM solvers can preserve total kinetic energy effectively due to inherent numerical diffusion, which is particularly noticeable in `rhoCentralFoam`. The results given from `rhoEnergyFoam` show that it conserves total kinetic energy under Mode A operation, aligning with theoretical expectations for energy-preserving schemes.

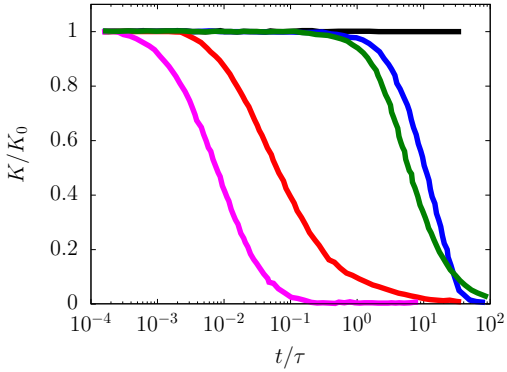
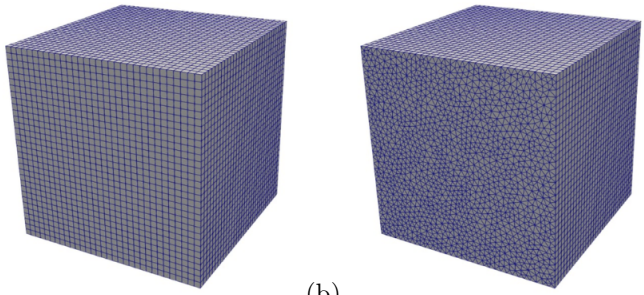


Figure 4.2. Decaying isotropic turbulence: time evolution of turbulence kinetic K energy for rhoEnergyFoam in Mode A (black line), Mode B (blue line), Mode C with $\theta^* = 0$ (red line), for dnsFOAM (green line), and rhoCentralFoam (magenta line). τ is the eddy turnover time.

When the pressure diffusion term is introduced (Mode B), a degree of numerical diffusion is observed, albeit lower than in `dnsFoam`. Most kinetic energy is dissipated at the characteristic eddy turn-over time scale. In Mode C, where $\theta^* = 0$, numerical diffusion is exacerbated but remains less pronounced than in `rhoCentralFoam`.

4.2 Taylor-Green Vortex

The Taylor–Green vortex solution is a classical case used for testing and validating spatial and temporal accuracy of CFD solvers. In fact, the Taylor-Green vortex illustrates the fundamental process of decaying production of small vortices, known as eddies.The energy-preserving properties of `rhoEnergyFoam` are herein tested checking the time reversibility of this test case. Tests were conducted on two types of meshes: a structured one (figure (4.3.a)) with $32'768$ cubic cells and a unstructured one (figure (4.3.b)) with $85'056$ prisms. Both of them are a $(2\pi)^3$ triply-periodic box generated with Gmsh (an open source finite-element mesh generator). The unstructured mesh was made extruding the triangular elements of the frontal face along the third dimension. This setting guarantees exact geometrical correspondence of the elements on opposite faces of the computational box, hence periodicity can be exploited in all space directions.



(a) (b)

Figure 4.3. Structured mesh (a) and unstructured mesh (b)for numerical simulation of Taylor–Green flow.

To verify the conservation of Turbulent Kinetic Energy (TKE), a time reversal test was performed in the middle of the simulation to ensure that the solution returned to the values of the initial conditions. The solution was initialized as follows:

$$\begin{cases} u = u_0 sin(k_0 x) cos(k_0 y) cos(k_0 z) \\ v = -v_0 cos(k_0 x) sin(k_0 y) cos(k_0 z) \\ w = 0 \end{cases} \tag{4.5}$$

where the density is constant $\rho = \rho_0$, $u_0 = Ma\ c_0$ is the reference velocity with $Ma = 0.01$ and c_0 is the reference speed of sound, P_0 is the reference pressure and T_0 is the reference temperature. The pressure equation used to initialize the case was the following:

$$p = p_0 + \frac{u_0^2}{16}[cos(2k_0 x) + sin(2k_0 y)][2 + cos(2k_0 z)] \tag{4.6}$$

where $k_0 = 1$ is the initial wave number. The initial conditions on the velocity field represented with the Q-criterion are show in figure (4.4)

The solution is evolved over time until $tu_0 k_0 = 8$, where all velocity vectors are reversed, and then continued further until $tu_0 k_0 = 16$. According to the mathematical principles governing the Euler equations, the initial conditions are expected to be precisely recovered.
Herein, we introduce a new physical variable called Enstrophy, defined as

$$\Omega = \frac{1}{2} \sum_i (\omega_k \omega_k)_i V_i. \tag{4.7}$$

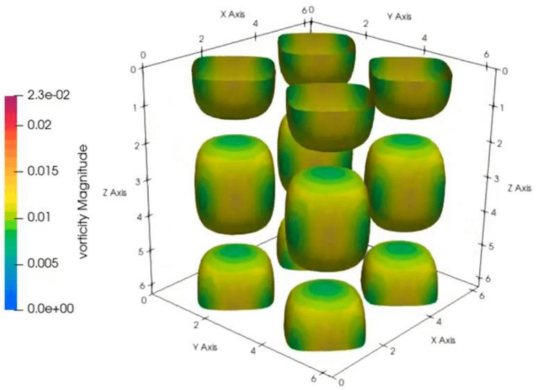


Figure 4.4. Vorticity field visualized using the Q-criterion for the Taylor-Green Vortex.

where $\omega = \nabla \times u$ represents the vorticity of the cell i and V_i is the volume of cell. Enstrophy is a measure of the intensity of vorticity in a moving fluid. In turbulence theory it provides insights into the distribution and dissipation of rotational motion within a flow. In inviscid flows, enstrophy can be conserved, whereas in viscous flows, it tends to dissipate over time due to the effects of viscosity. The evolution in time of the turbulent kinetic energy and of the enstrophy can be seen in figure (4.5). As illustrated in panel (a), the kinetic energy obtained from the simulation on the structured mesh using `rhoEnergyFoam` in both mode A and mode B remains conserved and constant, as indicated by the black solid line and the blue dotted line. In contrast, `rhoCentralFoam` exhibits a rapid dissipation of kinetic energy over a very short timescale, with curves that are nearly imperceptible in the adopted representation. Meanwhile, `dnsFoam` demonstrates a gradual decay of kinetic energy. In panel (b), which corresponds to the unstructured mesh, the total kinetic energy obtained using `dnsFoam` exhibits a continuous decay, whereas `rhoCentralFoam` continues to show rapid dissipation of kinetic energy. Conversely, `rhoEnergyFoam` effectively preserves kinetic energy in mode A, while mode B experiences some degree of numerical diffusion. In the Cartesian grid (panel c), the total enstrophy experiences a significant increase up to the point of time reversal, after which it symmetrically decreases. However,

with `dnsFoam`, the initial conditions are not perfectly restored, as the maximum vorticity at the conclusion of the simulation exceeds the expected value. This discrepancy arises from the randomization of the flow at the end of the direct phase, which is not entirely recovered in simulations influenced by numerical diffusion. On the unstructured grid (panel d), a similar trend is observed. Indeed, `rhoEnergyFoam` in mode A (black line) retains initial conditions, while `rhoEnergyFoam` in mode B shows a lower peak due to errors associated with mesh distortion and to the introduction of diffusion. In both cases, `dnsFoam` and `rhoCentralFoam` are not able to show good results.

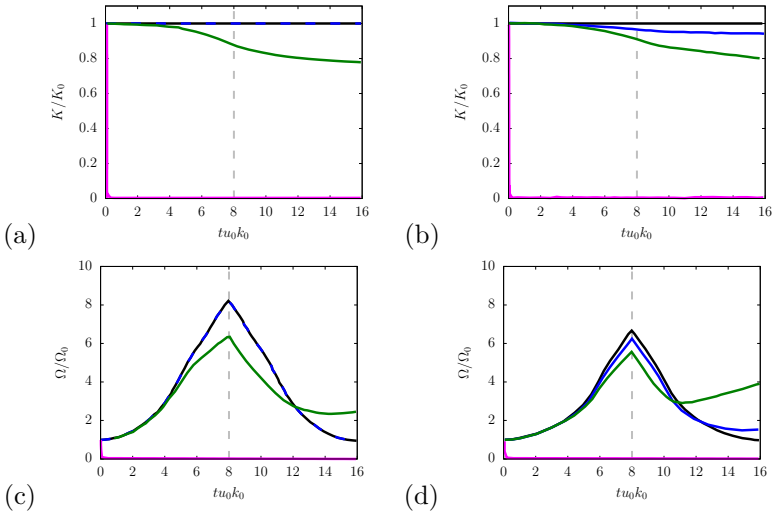


Figure 4.5. Taylor Green Vortex: time evolution of total kinetic energy (a-b), and enstrophy (c-d) for Taylor–Green flow at $Ma = 0.01$ on Cartesian mesh (a–c) and on unstructured mesh (b–d). Data are shown for rhoEnergyFoam in Mode A (black line), Mode B (blue dotted line), for dnsFOAM (green line) and rhoCentralFoam (magenta line).

4.3 Smooth flow past cylinder with AMR

The subsonic flow around a circular cylinder is here analyzed to check whether the upgrade of `rhoEnergyFoam` with the Adaptive Mesh Refinement (AMR) schemes (already present within OpenFOAM) works as expected. The simulation is performed on a 2D circular

cylinder at Mach $M_\infty = u_0/c_0 = 0.1$, where $c_0 = \sqrt{\gamma R T_0}$ is the adimensionalized speed of sound, $R = 1$ is the adimensionalized Gas constant and $T_0 = 1$ is the non-dimensional initial temperature. u_0 is the initial velocity field obtained with the previous equation, knowing the value of c_0 and of M_∞. The simulation is set using non-dimensional quantities such as the Reynolds number $Re_\infty = 100$, which is defined as

$$Re = \frac{\rho_\infty u_0 D}{\mu_\infty} \tag{4.8}$$

where ρ_∞ is the density of the fluid, D is the cylinder diameter, and μ_∞ is the dynamic viscosity. Also, the boundary of the domain used are $L_x = 14D$, $L_z = 8D$. The variables used are shown in table 4.1.

Variable	Value
M_∞	0.1
Re_D	100
ρ_∞	1
u_∞	0.118321
c_∞	1.18321
μ_∞	1.18321×10^{-4}
D	0.1

Table 4.1. Initial conditions for turbulent flow around circular cylinder.

The carthesian mesh, having 14′196 cells, is generated through the `blockMesh` utility, then it has been refined through the `snappyHexMesh` utility, as shown in figure (4.6).

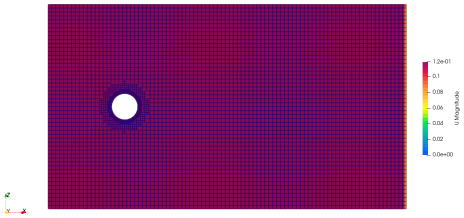


Figure 4.6. Smooth flow past cylinder with AMR: initial velocity field, with the mesh refinement only around the cylinder.

In figure (4.6), the initial velocity field is displayed, showing a uniform inflow condition and uniform velocity distribution around the cylinder.

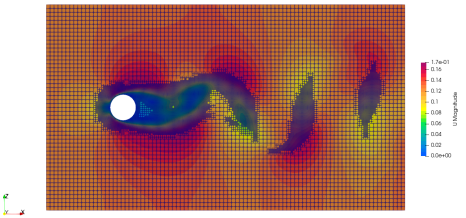


Figure 4.7. Smooth flow past cylinder with AMR: velocity field which shows the vortex shedding and the Kármán vortex street possible through the refinement of the mesh in the necessary zones.

In figure (4.7), the velocity field illustrates the onset of vortex shedding, characterized by alternating vortices in the wake of the cylinder. This phenomenon is known as vortex shedding an its pattern is typical for flows in the moderate Reynolds number regime, where the wake becomes unsteady and leads to the formation of the well-known Von Kármán vortex street.

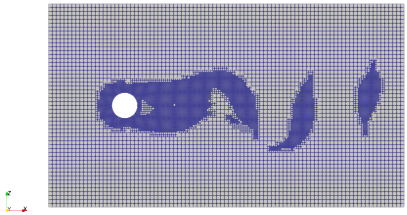


Figure 4.8. Cylinder mesh AMR.

Overall, the results highlight the transition from an initial steady-state condition to an unsteady wake due to flow separation and periodic vortex shedding.

4.4 URANS and DES of a flow past circular cylinder

The turbulent flow around a circular cylinder has been numerically investigated using *rhoEnergyFoam* in mode B, employing both URANS

(Unsteady Reynolds-Averaged Navier-Stokes) and DES (Detached Eddy Simulation). The study is based on the classic Spalart-Allmaras turbulence model and its DES extension. The free-stream Mach number is given by $M_0 = u_0/c_0 = 0.1$, where u_0 and c_0 are the free-stream velocity and the speed of sound. The Reynolds number, based on the cylinder diameter, is defined as $Re_D = (\rho_0 u_0 D)/\mu_w$, where ρ_0 represents the free-stream density and μ_w the wall viscosity. For the URANS simulation, an O-type mesh is utilized, consisting of $N_r \times N_\theta = 256 \times 256$ cells, with a hyperbolic tangent stretching towards the wall in a computational domain of $L_r \times L_z = 20D \times 2D$, as shown in figure 4.9. For the DES simulation, an also an O-type mesh is utilized, consisting of $N_r \times N_\theta \times N_z = 256 \times 256 \times 48$ cells. Here as well, a hyperbolic tangent stretching towards the wall was used, within the same domain as written above. The grid is refined near the cylinder, with the first grid point located at $y_+ \approx 150 - 200$, necessitating the use of wall functions for proper wall treatment. Specifically, Spalding's equilibrium wall law [42] is employed.

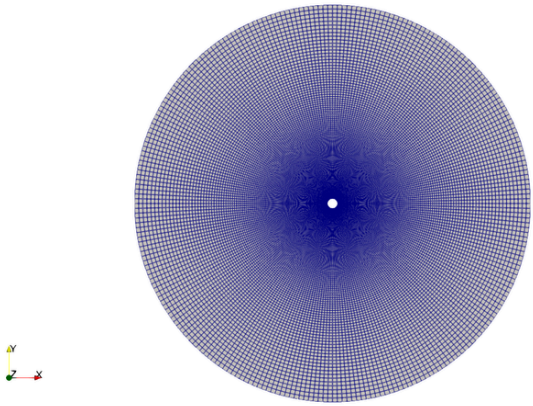


Figure 4.9. Mesh for the URANS ($N_r \times N_\theta \times N_z = 256 \times 256$) and DES ($N_r \times N_\theta \times N_z = 256 \times 256 \times 48$) simulation of the flow over a circular cylinder.

Isothermal no-slip boundary conditions are imposed at the wall, while `fixedValue` and/or `zeroGradient` boundary conditions are applied to all variables at the inlet and outlet, respectively. The turbulent viscosity is set to $\mu_{t0} = 3\mu_w$. The simulation has been performed in parallel, dividing the computational domain within 48 processors,

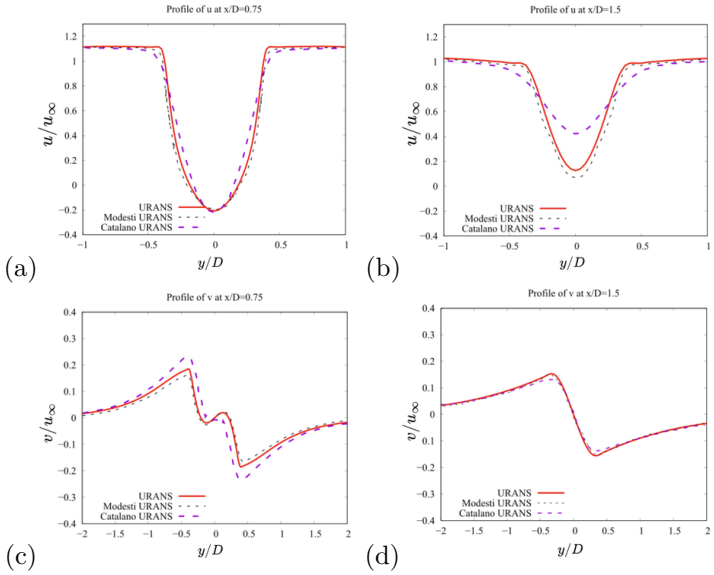


Figure 4.10. Numerical simulation of flow around circular cylinder: mean velocity profiles at $x/D = 0.75$ (panels a-c) e $x/D = 1.5$ (panels b-d) for URANS with `rhoEnergyFoam` (red line), Modesti's URANS (black dotted line) Catalano's URANS (purple dotted line).

in an HPC architecture. Numerical results are compared against previous simulations and experimental data. In figure 4.10 we present mean velocity profiles in the cylinder wake, whose results are of good acceptance with previous studies, with the only discrepancy that there is a longer wake in URANS compared to the reference numerical simulations by Catalano [3]. This deviation is likely caused by an inaccurate prediction of the separation point due to the approximate wall treatment. A key difference observed in URANS is the absence of significant vortex shedding, likely due to the use of wall functions. In contrast, vortex shedding is captured in the DES simulation, as shown in figure (4.13), with global flow parameters in reasonable agreement with reference sources. Further analysis of the wall pressure coefficient for the URANS and DES is presented in figures (4.11, 4.12). The overall comparison is satisfactory.

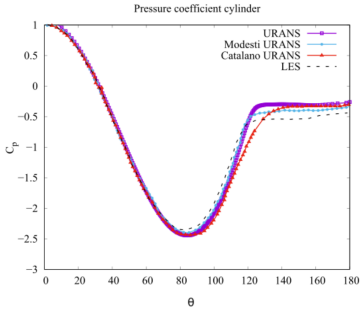


Figure 4.11. Numerical simulation of flow around a 2D circular cylinder: wall pressure coefficient obtained from rhoEnergyFoam in Mode B with URANS (purple line) Modesti's URANS (light blue line) and Catalano's URANS and LES (black dotted line).

4.5 Sod's shock tube

In this case, we simulate the Sod's shock tube problem, named after Gary A. Sod. This is a common test for the accuracy of computational fluid codes, like Riemann solvers, and was heavily investigated by Sod [39] in 1978. It consists of a pipe with either a circular or rectangular cross-section, filled with gas. The pipe is divided into two regions by a diaphragm positioned at its midpoint, as illustrated in figure (4.14). In numerical simulations, this diaphragm is represented as a discontinuity in the initial fluid conditions, including density, pressure, and temperature, across the interface. This is an extremely severe case used to test the accuracy and stability of CFD solvers. The initial conditions for this problem are straightforward, consisting of a contact discontinuity that separates gas regions with different pressure and density, while the velocity remains zero throughout the domain. In the standard configuration, the left side of the diaphragm has a density and pressure of 1, whereas the right side has a density of 0.125 and a pressure of 0.1. The specific heat ratio is set to 1.4.
These initial conditions can be expressed mathematically as follows.

$$\begin{bmatrix} \rho_2 \\ P_2 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.0 \end{bmatrix}, \quad \begin{bmatrix} \rho_1 \\ P_1 \\ u_1 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.1 \\ 0.0 \end{bmatrix} \tag{4.9}$$

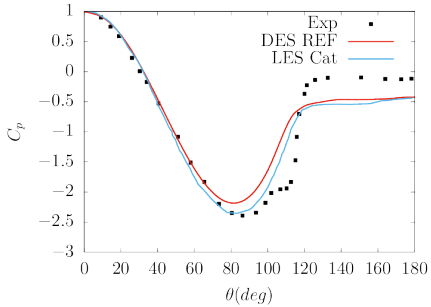


Figure 4.12. Numerical simulation of flow around a 3D circular cylinder: wall pressure coefficient obtained from rhoEnergyFoam in Mode B with DES (red line) and Catalano's DES (light blue line).

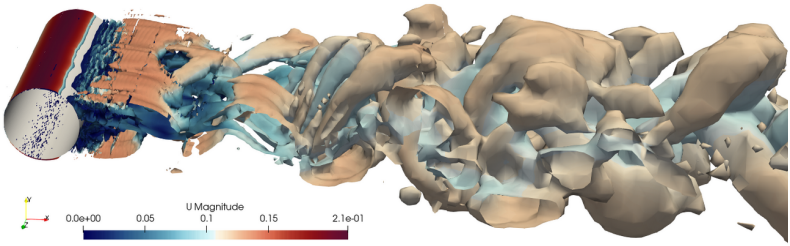


Figure 4.13. DES of a flow over a 3D circular cylinder: velocity filed plotted with Q-criterion $Q = \frac{1}{2}[(\nabla \cdot u)^2 - \nabla u : \nabla u^T]$.

where ρ is density, P is pressure and u is velocity. Index (2) and (1) refer to left and right-hand side conditions on the side of the diaphragm, respectively. The equations to be solved are the Euler equations, the air is modeled as an ideal gas and, in this case, `rhoEnergyFoam`0 is used in Mode C, where full AUSM 2.4.3 diffusion is activated near discontinuities to ensure stability and accuracy. The results are for similar runtime parameters (CFL, time step), accuracy and mesh.
Figure (4.15) contains six plots comparing the results of `rhoEnergyFoam` and `rhoCentralFoam`, for the Sod's shock tube simulation. Plots (a) and (b) depict the velocity magnitude, (c) and (d) show the pressure distribution, while (e) and (f) present the density profile. The red

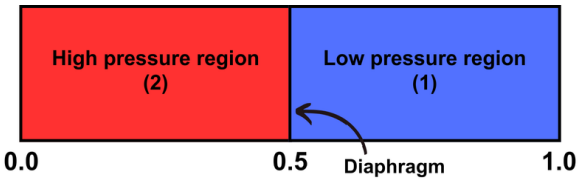


Figure 4.14. Sod shock tube at $t = 0$. On the left-hand side region with high density, pressure and temperature

dots represent the numerical results obtained from the solvers, while the black solid lines correspond to the exact analytical solution. From the comparison, it is evident that `rhoEnergyFoam` (plots a, c, and e) closely matches the exact solution with minimal deviation. In fact, it manages to capture the shock wave, rarefaction wave, and contact discontinuity accurately as well as the pressure drop across the shock and the density jump. In contrast, `rhoCentralFoam` (plots b, d, and f) exhibits noticeable discrepancies, particularly in the velocity and pressure profiles, where numerical oscillations appear around the discontinuities. This proves that `rhoEnergyFoam` provides a more accurate and stable solution for this type of problem, while `rhoCentralFoam` struggles to capture the shock and rarefaction waves correctly, becasue it exhibits oscillations around the shock and contact discontinuity.

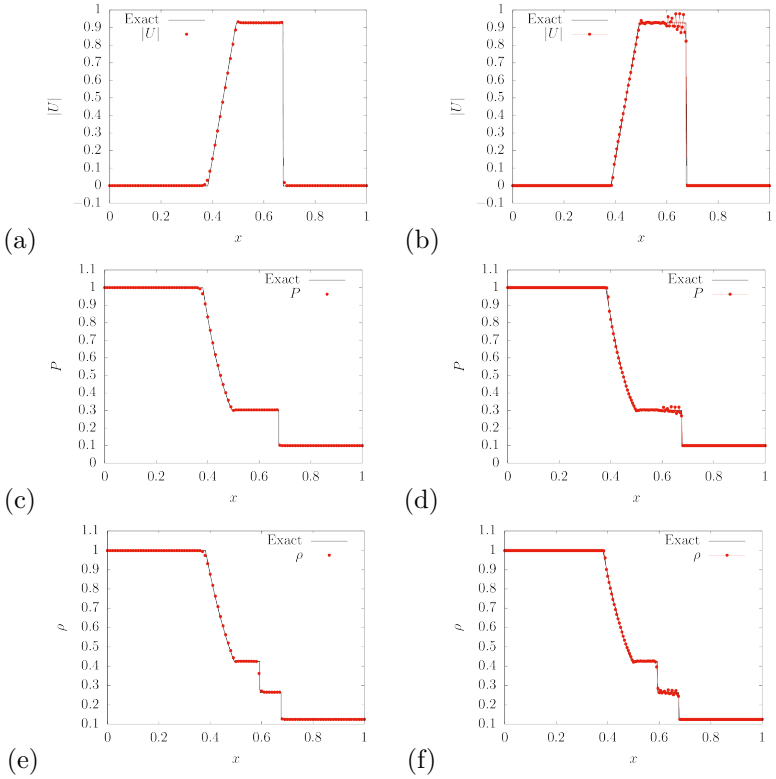


Figure 4.15. Sod's shock tube: Comparison of numerical results from `rhoEnergyFoam`(panels (a),(c),(e)) and `rhoCentralFoam` (panels (b),(d),(f)) for Sod's shock tube problem. The black solid lines represent the exact analytical solution, while the red dot show the numerical results. (a) and (b) display the velocity magnitude, (c) and (d) show the pressure distribution, and (e) and (f) present the density profile.

4.6 Supersonic flow over a forward-facing step

The study of inviscid flow over a forward-facing step was originally introduced by Emery [10] to evaluate the effectiveness of shock-capturing schemes. In this work, this configuration is used to analyze the performance of `rhoEnergyFoam` in simulating supersonic flows in the presence of shock waves.

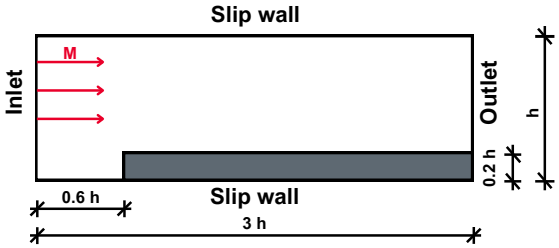


Figure 4.16. Geometry of the forward-facing step and its boundary conditions

The case considered involves a supersonic flow (Mach number $Ma = 3$) encountering a step of height $0.2\,h$, where h represents the channel height, as shown in figure 4.16. The total channel length is $3\,h$, with the leading edge of the step positioned at $0.6\,h$ from the inlet. The spatial discretization is uniform, consisting of a grid with 240×80 cells along the coordinate directions. The thermodynamic properties of the gas considered are defined as follows: specific heat at constant volume: $C_v = 1.785$ J/kgK, the gas constant: $R = 0.714286$ J/kgK and thermal conductivity: $k = 32.3\,\mu$W/mK. The boundary conditions include slip walls on the upper and lower surfaces, while all variables are extrapolated at the outlet.

In detail:

- **Inlet** (left): fixed-value condition for velocity $U = 3$ m/s (Mach 3), pressure $p = 1$ Pa, and temperature $T = 1$ K;
- **Outlet** (right): `zeroGradient` condition on U, p, and T;
- **Bottom wall**: no-slip adiabatic wall condition;
- **Top wall**: symmetry plane condition.

The initial conditions for the simulation are: velocity $U = 0$ m/s, pressure $p = 1$ Pa, and temperature: $T = 1$ K. The case is designed such that the speed of sound in the gas is $c = \sqrt{\gamma RT} = 1$ m/s. As a result, the velocity values are directly equivalent to the Mach number, meaning that an inlet velocity of 3 m/s corresponds to Mach 3.
This speed of sound relation can be verified using the equation for a perfect gas which yields the specific heat ratio $\gamma = \frac{C_p}{C_v}$. For this test case the solver is run in Mode C, with threshold value of the shock sensor $\theta^* = 0.05$ and $k_u = 0.25$, rather than the value suggested by Liou [23], as we observed a smoother shock front in this case. Analysis of the shock structure reveals that, although `rhoEnergyFoam` and `rhoCentralFOAM` produce qualitatively similar results as shown in figures (4.18, 4.17), `rhoEnergyFoam` provides additional flow details. In particular, the slip line emerging from the quadruple point near the upper wall is barely visible in `rhoCentralFOAM` due to its higher numerical diffusion.

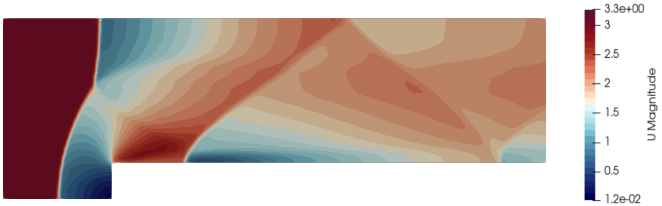


Figure 4.17. rhoCentralFoam: supersonic flow past forward-facing step at $Ma = 3$.

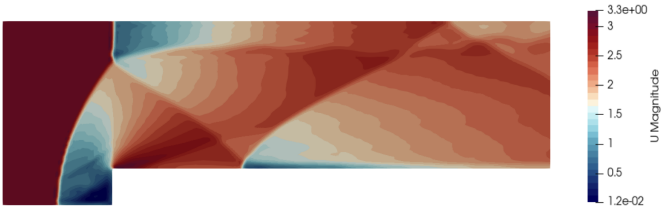


Figure 4.18. rhoEnergyFoam: supersonic flow past forward-facing step at $Ma = 3$.

4.7 Transonic flow over RAE-2822 airfoil

The RAE-2822 airfoil is herein simulated by Reynolds-Averaged Navier-Stokes (RANS) equations, with both `rhoEnergyFoam` in mode C and `rhoCentralFoam`, using the standard Spalart-Allmaras model. The flow is transonic with a Mach number $M = 0.729$ and the Reynolds number of the chord considered is $Re_c = 6.5 \times 10^6$, defined as

$$Re_c = \frac{\rho_0 u_0 c}{\mu_0} \tag{4.10}$$

where ρ_0 is the density, u_0 is the initial velocity c is the airfoil chord and μ_0 is the dynamic viscosity. The angle of attack used to simulate the airfoil is $\alpha = 2.31°$. A C-type structured mesh comprising 369×256 cells was used, with hyperbolic tangent extending toward the wall. The "farfield" boundary is located about 20 chords from the wall, where inlet/outlet boundary conditions are imposed, while isothermal nonslip boundary conditions are imposed on the airfoil wall.

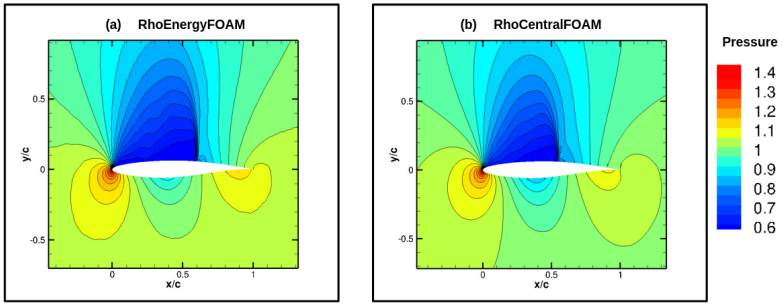


Figure 4.19. RANS of a transonic flow over RAE2822: Pressure fields evaluated with `rhoEnergyFoam` (a) and `rhoCentralFoam` (b). In the representation, 24 levels were used in the range $0.6 < p/p_\infty < 1.4$.

The calculated pressure fields are compared in figure (4.19), which shows the presence of a single normal shock on the suction side and minimal differences between the two solvers. The detailed comparison of the pressure coefficients with the Cook et al. [6] experiments and the [37] simulations, shown in figure (4.20), is satisfactory for both solvers, although in this case `rhoCentralFoam` seems to be closer to the experiments and `rhoEnergyFoam` closer to the previous simulations.

Finally we present table 4.2 which shows the lift and drag coefficient predicted by the two solver.

Case	C_l	C_d
rhoEnergyFoam	0.715	0.0135
rhoCentralFoam	0.730	0.0184
RANS [5]	0.717	0.0133
Experiment [6]	0.743	0.0127

Table 4.2. Lift and drag coefficient of RAE 2822 airfoil as predicted by rhoEnergyFoam and rhoCentralFoam, compared with Coakley [5] RANS and experimental data

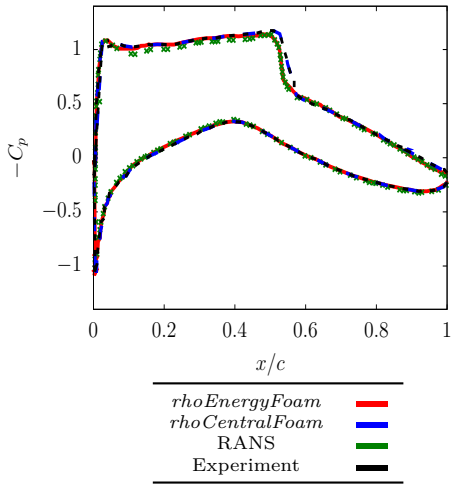


Figure 4.20. RANS of a transonic flow over RAE2822: pressure coefficient obtained with `rhoEnergyFoam` (red line) and with `rhoCentralFoam` (blue dotted lines),compared with previous RANS (green crosses) and experimental data (black dotted lines).

4.8 Transonic flow over Onera M6 wing

This case aims to study the flow around the wing of Onera M6, whose flow is inviscid ($\mu_0 = 0$), transonic with Mach number $M = 0.8395$, and the angle of attack of the wing is $\alpha = 3.06^{\circ}$. An unstructured mesh including 341'797 tetrahedral cells was used, within an external computational box of size $L_x = 10c$, $L_y = 10c$ and $L_z = 5c$, where c is the chord in the wing root section, as shown in figure (4.21). Numerical simulations were carried out using both `rhoCentralFoam` and `rhoEnergyFoam` in mode C, also comparing the results with experimental data values.

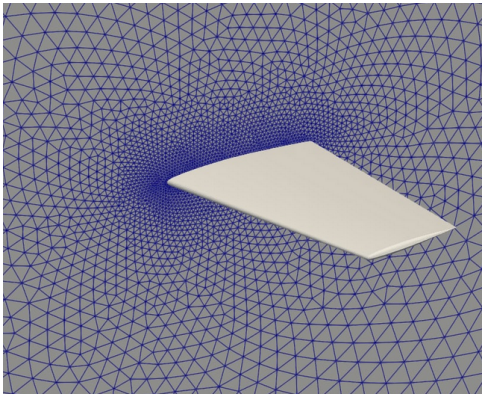


Figure 4.21. Unstructured mesh of the Onera M6 wing, within an external computational box of size $L_x = 10c$, $L_y = 10c$ and $L_z = 5c$, where c is the chord in the wing root section.

Numerical simulations were carried out using `rhoCentralFoam` and `rhoEnergyFoam` in mode C, also comparing the results with experimental data values. Figure (4.22) shows the pressure field obtained with `rhoEnergyFoam`, plotting the isosurface of the shock sensor over it, showing the presence of two shock waves: the main shock about halfway up the chord and a secondary shock near the leading edge, which eventually join near the wing tip.
Figure (4.23) shows the calculated pressure field on the wing suction surface for `rhoEnergyFoam` (figure 4.23a) and rhoCentralFoam (figure 4.23b), which highlights the qualitative differences between the two solvers.
Although the main flow characteristics are captured by both of them,

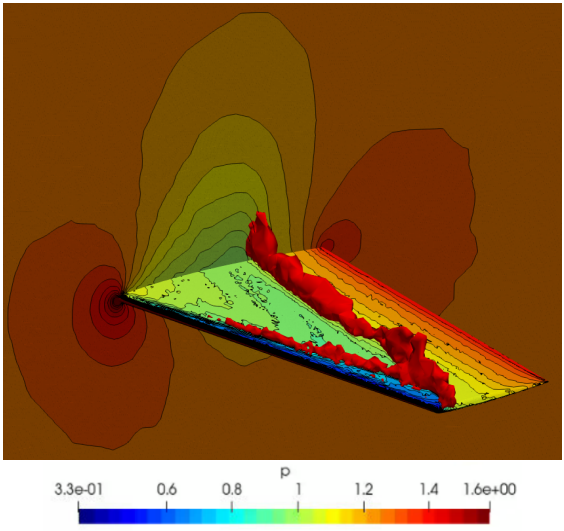


Figure 4.22. Pressure field with Ducros sensor isosurface superimposed ($\theta = 0.6$), showing the presence of two shock waves, of the Onera M6 wing. In the representation, 24 levels were used in the range $0.33 < p/p_\infty < 1.6$.

it appears that the edge shock is much weaker in rhoCentralFoam and the primary shock is much thicker, especially toward the wing root, due to the diffusive nature of the solver. A more quantitative assessment is made in figure (4.24), where the calculated distributions of the pressure coefficient are compared with Schmitt [36] experimental data, at the three wing sections indicated by the dashed lines in Figure (4.23). In the innermost section (panel a) the primary shock is rather weak and barely noticeable in `rhoCentralFoam`, while `rhoEnergyFoam` provides a favorable prediction of both the strength and position of the shock. In the second section (panels b) there are both shocks, which are again correctly captured by `rhoEnergyFoam`, while `rhoCentralFoam` shows excessive smearing. In the outermost section (panel c) the primary and secondary shocks merge into a single stronger shock, the amplitude of which is well captured by `rhoEnergyFoam`.

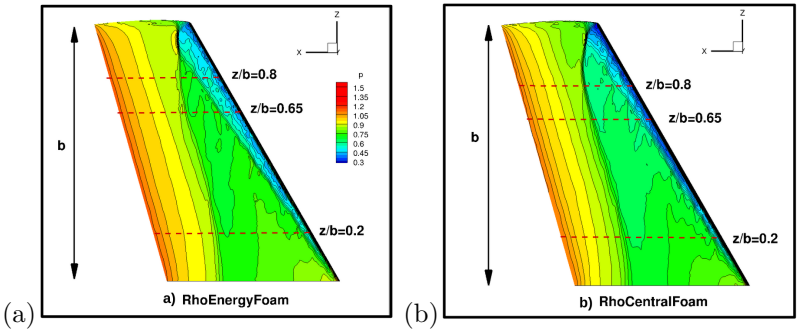


Figure 4.23. Flow around the wing of ONERA M6: pressure profiles evaluated with `rhoEnergyFoam` (a) and `rhoCentralFoam` (b). In range $0.3 < p/p_\infty < 1.5$ are shown 32 levels (blue-to-red scale). Dotted lines point to section used to evaluate pressure coefficients as in 4.24

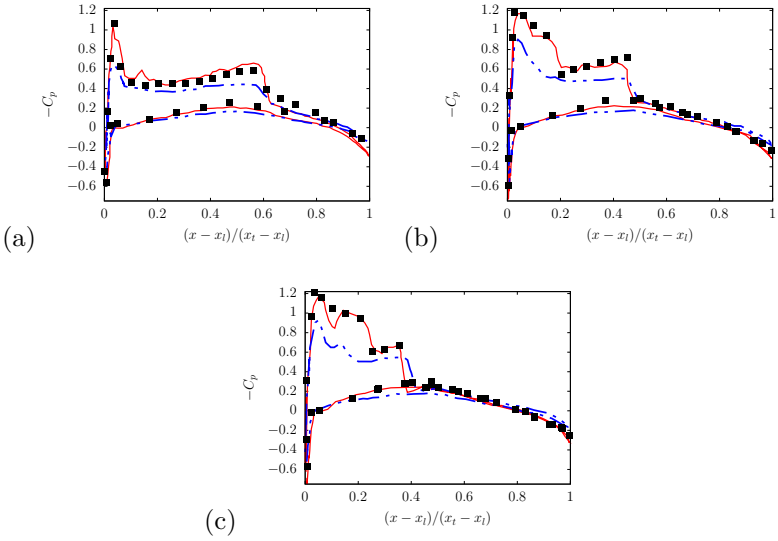


Figure 4.24. Flow around ONERA M6 wing: pressure coefficient ($Cp = (p - p_\infty)/(\frac{1}{2}\rho_\infty u_\infty^2)$) at various wing sections: (a) $z/b = 0.2$, (b) $z/b = 0.65$ and (c) $z/b = 0.8$, for rhoEnergyFoam (red lines), rhoCentralFoam (blue dotted lines) and experimental data (black squares). x_l and x_t denote the coordinates of the leading edge and trailing edge of each wing section, respectively.

Chapter 5

Conclusion and Future Development

The development of the `rhoEnergyFoam` solver represents a significant advancement in the numerical simulation of compressible turbulent flows. By leveraging an energy-consistent, non-diffusive numerical flux and selectively incorporating diffusion based on flow characteristics, the solver effectively balances accuracy and stability. The introduction of a triple splitting approach, previously unexplored in finite-volume methods, and the decomposition of the AUSM diffusive flux into distinct pressure and convective components ensure minimal numerical dissipation while maintaining robustness. The three-mode operational framework provides tailored numerical diffusion for different flow regimes: fully resolved simulations (Mode A) operate without artificial diffusion, unresolved smooth flows (Mode B) utilize minimal pressure diffusion, and shocked flows (Mode C) employ both pressure and convective diffusion to enhance stability.

Implemented within the OpenFOAM framework, `rhoEnergyFoam` has demonstrated superior performance across a spectrum of test cases, from canonical turbulence problems to applied aerodynamic flows. It significantly outperforms standard OpenFOAM solvers in preserving kinetic energy, a crucial feature for unsteady RANS and Large Eddy Simulation (LES), thereby improving turbulence accuracy and aerodynamic force predictions. This reduction in numerical diffusion ensures that turbulence structures are preserved without unnecessary dissipation, addressing a common drawback of traditional schemes.

The ability to retain energy fidelity in turbulence simulations is essential for accurately capturing flow physics, making `rhoEnergyFoam` a valuable tool for high-resolution CFD applications.
A key advantage of `rhoEnergyFoam` is its ability to accurately capture shock phenomena while minimizing numerical diffusion. By selectively applying AUSM pressure and convective diffusive fluxes based on a local shock sensor, the solver avoids excessive dissipation, ensuring sharp resolution of shocks while maintaining stability. This capability is particularly relevant for aerospace and automotive applications, where capturing shocks in supersonic flows simulations is crucial for aerodynamic performance. Traditional solvers often struggle to achieve this balance, leading to either excessive numerical damping that compromises accuracy or unstable solutions that necessitate additional artificial dissipation. `rhoEnergyFoam` mitigates these issues by dynamically adjusting diffusion, allowing it to operate efficiently across a broad range of Mach and Reynolds numbers.
The open-source nature of the solver fosters collaboration and further development within the CFD community. Open-source accessibility encourages customization, improvement, and widespread adoption, bridging the gap between specialized high-accuracy numerical methods and general-purpose CFD solvers. This makes `rhoEnergyFoam` a valuable tool for both academic research and industrial applications. By providing an accessible platform for innovation, the solver enables users to develop new numerical techniques, optimize turbulence models, and expand its applicability to complex engineering challenges.
Moreover, the solver's ability to maintain accuracy on unstructured meshes without requiring excessive artificial diffusion enhances its usability in industrial settings, where complex geometries and varying grid resolutions are common. The incorporation of low-dissipative numerics ensures that intricate flow structures are retained even in highly turbulent environments. This feature is particularly useful in engineering disciplines such as aerodynamics, propulsion, and energy systems, where accurate flow predictions are essential for optimizing design and performance.
In conclusion, `rhoEnergyFoam` provides a powerful, low-dissipation, energy-preserving approach for high-fidelity CFD simulations. Its integration within OpenFOAM enhances accessibility, while its innovative numerical strategies improve accuracy and stability across a wide range of flow problems. Through continued development and adoption, `rhoEnergyFoam` has the potential to reshape computational fluid dynamics, enabling more efficient and precise simulations for

engineering applications. As a future step, GPU acceleration could significantly enhance computational efficiency, reducing turnaround times for large-scale simulations and further broadening its industrial applicability. Implementing a GPU-accelerated version of `rhoEnergyFoam` would allow for larger simulations to be performed at unprecedented speeds, facilitating real-time analysis and optimization in engineering workflows. This advancement would make high-fidelity simulations more practical for time-sensitive applications, reinforcing the solver's role as a next-generation CFD tool.

Bibliography

[1] N. A. Adams and K. Shariff. A high-resolution hybrid compact-eno scheme for shock-turbulence interaction problems. *Journal of Computational Physics*, 127(1):27–51, 1996. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1996.0156.

[2] J. Boussinesq. *Essai sur la théorie des eaux courantes*. Mémoires présentés par divers savants à l'Académie des sciences de l'Institut national de France. Impr. nationale, 1877. URL `https://books.google.it/books?id=4NdQAAAAYAAJ`.

[3] P. Catalano, M. Wang, G. Iaccarino, and P. Moin. Numerical simulation of the flow around a circular cylinder at high reynolds numbers. *International journal of heat and fluid flow*, 24(4): 463–469, 2003.

[4] R. W. Clough. Original formulation of the finite element method. *Finite elements in analysis and design*, 7(2):89–101, 1990.

[5] T. Coakley. Numerical simulation of viscous transonic airfoil flows. In *25th AIAA Aerospace Sciences Meeting*, page 416, 1987.

[6] P. H. Cook, M. C. P. Firmin, and M. A. McDonald. *Aerofoil RAE 2822: pressure distributions, and boundary layer and wake measurements*. RAE, 1977.

[7] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical proceedings of the Cambridge philosophical society*, volume 43, pages 50–67, 1947. doi: 10.1017/S0305004100023197.

[8] F. Darwish, L. Moukalled, and M. Mangani. *The finite volume method in computational fluid dynamics*. Springer, 2016.

[9] F. Ducros, V Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, and T. Poinsot. Large-eddy simulation of the shock/turbulence interaction. *Journal of Computational Physics*, 152(2):517–549, 1999. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1999.6238.

[10] A. F. Emery. An evaluation of several differencing methods for inviscid fluid flow problems. *Journal of Computational Physics*, 2(3):306–331, 1968.

[11] A. Favre. Turbulence: Space-time statistical properties and behavior in supersonic flows. *The Physics of Fluids*, 26(10): 2851–2863, 1983. ISSN 0031-9171. doi: 10.1063/1.864049.

[12] J. H. Ferziger, M. Perić, and R. L. Street. *Computational methods for fluid dynamics*, volume 113. Springer, 2016. doi: 10.1007/978-3-319-16874-6.

[13] Foundation. Openfoam, Accessed on November 11, 2024. URL `https://openfoam.org/`.

[14] Eric Garnier, Pierre Sagaut, and Michel Deville. A class of explicit eno filters with application to unsteady flows. *Journal of Computational Physics*, 170(1):184–204, 2001.

[15] A. D. Gosman. Heat and mass transfer in recirculating flows. *(No Title)*, 1969.

[16] A. Harten et al. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(3):357–393, 1983. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(83)90136-5.

[17] Charles Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

[18] R. I. Issa, A. D. Gosman, and A. P. Watkins. The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. *Journal of Computational Physics*, 62 (1):66–82, 1986. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(86)90100-2.

[19] A. Jameson. Origins and further development of the jameson–schmidt–turkel scheme. *AIAA Journal*, 55(5):1487–1510, 2017. doi: 10.2514/1.J055493.

[20] P. Lax and B. Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13(2): 217–237, 1960. doi: https://doi.org/10.1002/cpa.3160130205. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160130205`.

[21] B. P. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering*, 19(1):59–98, 1979.

[22] M. S. Liou. A sequel to ausm: Ausm+. *Journal of Computational Physics*, 129(2):364–382, 1996. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1996.0256.

[23] M. S. Liou. A sequel to ausm, part ii: Ausm+-up for all speeds. *Journal of Computational Physics*, 214(1):137–170, 2006. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2005.09.020.

[24] M. S. Liou and C. J. Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107(1):23–39, 1993. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1993.1122.

[25] X. D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1): 200–212, 1994. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1994.1187.

[26] D. Modesti and S. Pirozzoli. A low-dissipative solver for turbulent compressible flows on unstructured meshes, with openfoam implementation. *Computers & Fluids*, 152:14–23, 2017.

[27] G. Moretti. The λ-scheme. *Computers & Fluids*, 7(3):191–205, 1979. ISSN 0045-7930. doi: https://doi.org/10.1016/0045-7930(79)90036-7.

[28] OpenCFD. Openfoam, Accessed on October 29, 2024. URL `https://www.openfoam.com/`.

[29] S. V. Patankar. *Numerical heat transfer and fluid flow*. Series on Computational Methods in Mechanics and Thermal Science.

Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980. ISBN 978-0891165224.

[30] S. Pirozzoli. Conservative hybrid compact-weno schemes for shock-turbulence interaction. *Journal of Computational Physics*, 178(1):81–117, 2002. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2002.7021.

[31] S. Pirozzoli. Generalized conservative approximations of split convective derivative operators. *Journal of Computational Physics*, 229(19):7180–7190, 2010. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2010.06.006.

[32] S. Pirozzoli. Stabilized non-dissipative approximations of euler equations in generalized curvilinear coordinates. *Journal of Computational Physics*, 230(8):2997–3014, 2011. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2011.01.001.

[33] S. Pirozzoli. Numerical methods for high-speed flows. *Annual Review of Fluid Mechanics*, 43:163–194, 2011.

[34] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(81)90128-5.

[35] A. K. Runchal and M. Wolfshtein. Numerical integration procedure for the steady state navier-stokes equations. *Journal of Mechanical Engineering Science*, 11(5):445–453, 1969.

[36] V. Schmitt. Pressure distributions on the onera m6-wing at transonic mach numbers, experimental data base for computer program assessment. *AGARD AR-138*, 1979.

[37] J. W. Slater. Rae2822 transonic airfoil: Study# 1. *NASA Glenn Research Centre, Ohio. Internet: http://www. grc. nasa. gov/WWW/wind/valid/raetaf/raetaf01/raetaf01. ht ml*, 2010.

[38] J. Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[39] G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of*

Computational Physics, 27(1):1–31, 1978. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(78)90023-2.

[40] P. Spalart. Detached-eddy simulation. *Annual Review of Fluid Mechanics*, 41:181–202, 01 2009. doi: 10.1146/annurev.fluid.010908.165130.

[41] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA Paper 1992-0439*, page 439, 1992.

[42] D. B. Spalding et al. A single formula for the law of the wall. *Journal of Applied Mechanics*, 28(3):455–458, 1961.

[43] J. L. Steger and R. F. Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40(2):263–293, 1981. ISSN 0021-9991.

[44] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Science & Business Media, 2013.

[45] B. Van Leer and E. Krause. Lecture notes in physics, vol. 170, numerical methods in fluid dynamics. *SIAM review*, 1982.

[46] D. C. Wilcox. Formulation of the k-w turbulence model revisited. *AIAA Journal*, 46(11):2823–2838, 2008. doi: 10.2514/1.36541.

[47] H. C. Yee, N. D. Sandham, and M. J. Djomehri. Low-dissipative high-order shock-capturing methods using characteristic-based filters. *Journal of Computational Physics*, 150(1):199–238, 1999. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1998.6177.