Natalia Denton

Design of an automated OpenFOAM Workflow for External-Aerodynamics Simulations

School of Aerospace, Transport and Manufacturing
Computational Fluid Dynamics

MSc
Academic Year: 2024–2025

Supervisor: Dr Tsoutsanis Panagiotis
Aug 2025

School of Aerospace, Transport and Manufacturing
Computational Fluid Dynamics

MSc

Academic Year: 2024–2025

Natalia Denton

Design of an automated OpenFOAM Workflow for External-Aerodynamics Simulations

Supervisor: Dr Tsoutsanis Panagiotis
Aug 2025

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

© Cranfield University 2025. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

The central research question of this thesis is: *Can the aerodynamic simulation workflow for automotive applications be fully automated using open-source tools such as OpenFOAM, while maintaining accuracy and mesh quality suitable for engineering evaluation?* This question is significant in the automotive sector, where highly competitive development cycles demand rapid, cost-effective aerodynamic testing across multiple geometries. Traditional manual workflows limit the number of design iterations, whereas automation offers the potential to increase throughput while reducing human error and operational costs.

The workflow developed in this project integrates Python scripting to automatically generate OpenFOAM case files from YAML configuration inputs. The meshing process leverages `SnappyHexMesh`, and simulations were performed using `simpleFoam`, `potentialFoam`, and turbulence models including k-ε, k-ω, and laminar flows. Post-processing was conducted using `ParaFOAM`, allowing automated extraction of aerodynamic coefficients. This approach was successfully applied to three benchmark cases: the *AeroSUV*, *unitCube*, and *Ahmed Body*, demonstrating the feasibility of a fully automated simulation pipeline.

Key challenges were observed in the **meshing stage**, where `SnappyHexMesh` presented limitations in achieving consistently high-quality meshes, particularly in regions requiring refined boundary layers. In commercial workflows, tools like ANSA provide more robust control over mesh quality; however, their unavailability during this project necessitated fully open-source solutions. **As such, the developed workflow stands as a proof of concept:** it successfully automates the aerodynamic simulation process, reduces

manual setup time, and enables testing of multiple geometries, but further refinement of mesh control and integration with advanced tools would be required for production-level deployment.

Keywords:
workflow automation; computational fluid dynamics; turbulence modeling; mesh generation; open-source simulation; aerodynamic design; Python scripting; parametric studies; automated preprocessing; automotive aerodynamics

Acknowledgements

The author would like to express sincere gratitude to her supervisors, Dr. Panagiotis Tsoutsanis, Robin East, and Dr. Antonis Antoniadis, for their invaluable guidance, continuous support, and constructive feedback throughout this project. Their expertise in computational fluid dynamics and automotive aerodynamics has been instrumental in the successful completion of this research.

The author would also like to acknowledge McLaren Automotive for providing industrial insight and context, which greatly enhanced the practical relevance of this work. Finally, heartfelt thanks are extended to colleagues, friends, and family for their encouragement and support during the course of this thesis.

Table of Contents

Academic Integrity Declaration i

Abstract ii

Acknowledgements iv

Table of Contents v

List of Figures x

List of Tables xii

List of Abbreviations xiv

1 Introduction 1
- 1.1 Aim and Objectives . . . 2
- 1.2 Significance and Contribution . . . 2
- 1.3 Thesis Structure . . . 3

2 Literature Review 4
- 2.1 Introduction to the Literature Review . . . 4
- 2.2 Overview of CFD in Automotive Aerodynamics . . . 4
- 2.3 Mesh Generation and Geometry Preparation in CFD . . . 6
  - 2.3.1 Meshing Strategies . . . 6
    - 2.3.1.1 Structured Meshes . . . 7
    - 2.3.1.2 Unstructured Meshes . . . 7
    - 2.3.1.3 Polyhedral Meshes . . . 8
    - 2.3.1.4 Hybrid Meshes . . . 8
    - 2.3.1.5 Comparative Considerations . . . 9
    - 2.3.1.6 Summary . . . 9
  - 2.3.2 Meshing Tools in Literature . . . 9
    - 2.3.2.1 Commercial Meshing Tools . . . 9
    - 2.3.2.2 Open-Source Meshing Tools . . . 10
    - 2.3.2.3 Comparative Evaluation and Literature Insights . . . 10
  - 2.3.3 Mesh Quality and Validation . . . 11
    - 2.3.3.1 Mesh Quality Metrics . . . 11
      - 2.3.3.1.1 Skewness . . . 11

2.3.3.1.2 Orthogonality . . . 11
2.3.3.1.3 Aspect Ratio . . . 11
2.3.3.1.4 y^+ Calculation and Turbulence Model Requirements . . . 11
2.3.3.2 Effect on Solver Stability and Accuracy . . . 12
2.3.3.3 Boundary-Layer Inflation and Refinement . . . 12
2.3.3.4 Summary of Best Practices . . . 12
2.4 Turbulence Modeling in Automotive CFD . . . 13
2.4.1 RANS Models . . . 13
2.4.1.1 Governing Equations . . . 13
2.4.1.2 Eddy Viscosity Hypothesis . . . 13
2.4.1.3 The $k-\varepsilon$ Model . . . 13
2.4.1.4 The SST $k-\omega$ Model . . . 14
2.4.1.5 Summary . . . 14
2.4.2 LES and Hybrid Approaches . . . 14
2.4.3 Comparison of Models in Literature . . . 15
2.4.3.1 Generic SUV Benchmark . . . 15
2.4.3.2 DrivAer Notchback/Fastback Analysis . . . 16
2.4.3.3 Methodology Evaluation for Realistic SUV Flows . . . 16
2.4.3.4 Summary and Model Selection Justification . . . 16
2.5 CFD Solvers and Numerical Methods . . . 17
2.5.1 Governing Equations and Pressure–Velocity Coupling . . . 17
2.5.2 SIMPLE, PISO, and Fully Coupled Approaches . . . 17
2.5.3 Solver Selection in Literature . . . 18
2.6 Validation Studies and Benchmarks . . . 18
2.6.1 AeroSUV Body Studies . . . 18
2.6.2 Other Reference Geometries . . . 19
2.6.2.0.1 DrivAer Notchback, Fastback, and Estateback 19
2.6.3 Gaps in Current Literature . . . 19
2.7 Summary of Key Findings . . . 20

3 Methodology **21**
3.1 Workflow Overview . . . 21
3.2 Configuration System . . . 22
3.2.1 User-Level Configuration . . . 22
3.2.2 Advanced/Developer Configuration . . . 23
3.2.3 Configuration Philosophy . . . 24
3.2.4 Choice of Meshing Tool . . . 25
3.2.5 Meshing Procedure . . . 25
3.2.5.1 Step-by-Step Workflow . . . 26
3.2.5.2 Automation and Integration . . . 27
3.2.5.3 Integration of Mesh Quality Metrics with Solver Performance . . . 28
3.2.5.3.1 Skewness and Solver Stability . . . 28
3.2.5.3.2 Orthogonality and Numerical Diffusion . . . 28
3.2.5.3.3 Aspect Ratio and Boundary-Layer Resolution 29

3.2.5.3.4 y^+ and Turbulence Model Requirements . . . 29
3.2.5.3.5 Boundary-Layer Inflation and Adaptive Refinement . . . 29
3.2.5.3.6 Synthesis of Best Practices . . . 29
3.2.6 Challenges and Limitations . . . 30
3.3 Turbulence Modeling . . . 31
3.3.0.1 Laminar Flow Model . . . 31
3.3.0.2 Standard $k-\varepsilon$ Model . . . 32
3.3.0.3 SST $k-\omega$ Model . . . 32
3.3.0.4 Rationale for Enabling All Models . . . 32
3.4 Automation and Post-Processing . . . 33
3.4.1 Workflow Automation . . . 33
3.4.1.0.1 Example Shell Script Workflow . . . 33
3.4.1.0.2 Key Points: . . . 33
3.4.2 Dynamic OpenFOAM Case File Generation . . . 33
3.4.2.0.1 Advantages: . . . 34
3.4.3 Containerization and HPC Portability . . . 34
3.4.4 Post-Processing and Data Extraction . . . 34
3.5 Validation Setup . . . 35
3.5.1 Benchmark Geometries . . . 35
3.5.2 Boundary Conditions and Test Matrix . . . 35

4 Results and Discussion **37**
4.1 Introduction to Results . . . 37
4.2 Verification and Validation (V&V) . . . 38
4.2.1 Unit Cube . . . 38
4.2.1.1 Mesh Independence Study . . . 38
4.2.1.2 Continuity Residual Analysis by Mesh Density . . . 46
4.2.1.2.1 Coarse Mesh Analysis . . . 46
4.2.1.2.2 Medium Mesh Analysis . . . 46
4.2.1.2.3 Fine Mesh Analysis . . . 47
4.2.1.3 Force Coefficients by Mesh Density . . . 48
4.2.1.4 Grid Convergence Index (GCI) Assessment . . . 52
4.2.1.5 Computed Results and Literature Validation . . . 52
4.2.1.5.1 Interpretation of Results . . . 53
4.2.1.5.2 Literature Context . . . 53
4.2.1.5.3 Summary and Confidence Level . . . 54
4.2.1.6 Solver Convergence Study . . . 54
4.2.1.7 Solver Comparison . . . 55
4.2.1.7.1 Discussion . . . 56
4.2.1.8 Pseudo Time Step Analysis . . . 57
4.2.1.9 Flow Field Analysis . . . 58
4.2.1.9.1 Pressure Distribution . . . 59
4.2.1.9.2 Stagnation Point . . . 60
4.2.1.9.3 Low-Pressure Zones . . . 60
4.2.1.9.4 Comparison to Bluff-Body Patterns . . . 60

4.2.1.9.5 Velocity and Wake Structure 61
4.2.1.9.6 Separation and Recirculation Zones 61
4.2.1.9.7 Wake Symmetry 61
4.2.1.9.8 Comparison with Literature 61
4.2.1.9.9 Conclusions 62
4.2.1.10 Forces and Coefficients 63
4.2.1.10.1 Drag Decomposition 63
4.2.1.10.2 Side-Force Characteristics 63
4.2.1.10.3 Comparison with Expected Trends 63
4.2.1.10.4 Temporal Convergence and Residuals 64
4.2.1.10.5 Implications for Bluff-Body Modeling 64
4.2.1.11 Automation and Workflow Efficiency 64
4.2.1.12 Solver Performance and Computational Efficiency . . . 64
4.2.1.12.1 Mesh Generation Timings and Quality Metrics 64
4.2.1.12.2 Solver Timing for Mesh Resolutions 65
4.2.1.12.3 Time-Step Sensitivity on the Medium Mesh . 65
4.2.1.12.4 Solver Comparison on Fine Mesh 66
4.2.2 AeroSUV 66
4.2.2.1 Mesh Quality Analysis 66
4.2.2.1.1 Cell Type Composition 67
4.2.2.1.2 Layering Statistics 68
4.2.2.1.3 Refinement Level Distribution 69
4.2.2.1.4 Quality vs. Thresholds 70
4.2.2.1.5 Mesh Analysis summary 72
4.2.2.2 Continuity Residual Analysis 74
4.2.2.3 Force Coefficients 75
4.2.2.4 Flow Field Analysis 77
4.2.2.4.1 Pressure - Internal Fluid Pressure 78
4.2.2.4.2 Pressure - Surface Pressure 80
4.2.2.4.3 Velocity Fields 83
4.2.2.4.4 Vorticity 90
4.2.2.5 Solver Performance and Computational Efficiency . . . 92
4.2.2.5.1 Mesh Generation Timings and Quality Metrics 92
4.2.2.5.2 Solver Timing for Mesh Resolutions 93
4.2.3 Discussion of Key Findings 93
4.2.3.0.1 Most significant aerodynamic insights 93
4.2.3.0.2 Did the automated workflow meet accuracy and efficiency objectives? 94
4.2.3.0.3 Main limitations (mesh, turbulence model, solver) 94
4.2.3.0.4 Unexpected flow features and numerical artefacts 94
4.2.3.0.5 How results support feasibility of fully automated CFD workflows 95
4.2.3.0.6 Practical recommendations and next steps . . 95
4.2.3.0.7 Concluding remark 96

5 Conclusions **97**
5.0.1 Summary Of Findings . . . 97
5.0.1.1 mesh independence . . . 97
5.0.1.2 Validation . . . 97
5.0.1.2.1 Procedure and scaling . . . 97
5.0.1.2.2 Numbers used . . . 98
5.0.1.2.3 Computed forces and comparisons . . . 98
5.0.1.2.4 Discussion . . . 99
5.0.1.2.5 Conclusion of validation . . . 99
5.0.1.3 wake analysis . . . 99
5.0.1.4 Solver Performance and Computational Efficiency . . . 100
5.0.1.4.1 Mesh Generation Timings and Quality Metrics 100
5.0.1.4.2 Solver Timing for Mesh Resolutions . . . 100
5.1 Summary of Work . . . 100
5.2 Future Work . . . 100
5.2.1 Pre-Processing . . . 101
5.2.2 Meshing . . . 101
5.2.3 Solving . . . 102
5.2.4 Post-Processing . . . 102

A Ethical Approval Letter **113**

List of Figures

2.1 Structured mesh for a complex airfoil wing, illustrating the challenges of meshing intricate geometries (adapted from [27]). 7
2.2 Unstructured mesh for an automotive configuration (adapted from [28]). . 7
2.3 Polyhedral mesh used in an F1 aerodynamic application (adapted from [30]). 8
2.4 Hybrid mesh for an SUV aerodynamic study [31]. 8

3.1 Meshing pipeline using `snappyHexMesh`, showing the stages from CAD geometry to solver-ready mesh: background mesh generation, castellated cell cutting, surface snapping, and prism-layer addition. 28

4.1 Flow field for coarse mesh. 39
4.2 Wall y^+ distribution for coarse mesh. 40
4.3 Flow field for medium mesh. 41
4.4 Wall y^+ distribution for medium mesh. 42
4.5 Flow field for fine mesh. 43
4.6 Wall y^+ distribution for fine mesh. 44
4.7 Evolution of wall y^+ over time for coarse, medium, and fine meshes. . . . 45
4.8 Time history of force coefficients for the coarse mesh. The signal exhibits initial transients followed by stable oscillations. 49
4.9 Time history of force coefficients for the medium mesh. Oscillation levels are similar to the coarse mesh but show improved smoothness. 49
4.10 Time history of force coefficients for the fine mesh. Mean values remain stable and closely match coarser meshes, indicating mesh convergence. . 50
4.11 Time-averaged aerodynamic force coefficients between $t = 8$ s and $t = 10$ s with temporal standard deviation shown as black error bars. 51
4.12 Residual convergence of the velocity component U_x over 1000 solver iterations for coarse, medium, and fine meshes. All meshes achieve convergence near a residual magnitude of 10^{-4}. The medium mesh exhibits a sudden residual spike around iteration 700, which quickly decays. . . . 51
4.13 Richardson extrapolation of C_d with grid refinement and fine-grid GCI band. 53
4.14 Comparison of U_x residual histories for the `k-epsilon` and `k-omega SST` models. 54
4.15 Global continuity error convergence for both turbulence models. 55
4.16 Evolution of the average wall y^+ during the simulations. 55
4.17 Time history of force coefficients for the `k-epsilon` model. 56

4.18 Time history of force coefficients for the `k-omega SST` model. 56
4.19 Pressure distribution on the side face (x–constant plane) at $t = 20$ s. 59
4.20 Pressure distribution on the y–constant plane at pseudo time $t = 20$ s. . . . 59
4.21 Pressure distribution on the top face z–constant plane at pseudo time $t = 20$ s. 60
4.22 Velocity magnitude on the x–constant plane at pseudo time $t = 20$ s. . . . 61
4.23 Velocity magnitude on the y–constant plane at pseudo time $t = 20$ s. . . . 62
4.24 Velocity magnitude on the z–constant plane at pseudo time $t = 20$ s. . . . 62
4.25 Distribution of cell volumes across the computational mesh, visualised in ParaView using the `Cell Size` filter. Warmer colours indicate larger cells, cooler colours smaller cells. 69
4.26 Close-up of vehicle showing y^+ distribution at pseudo-time $t = 5$. 71
4.27 Side view showing patchiness in boundary layer formation and its effect on y^+. 71
4.28 Top view showing patchy layering and y^+ distribution in the wheel region. 72
4.29 Convergence history of continuity errors 74
4.30 Residual convergence of the velocity component U_x over 5000 solver iterations for the produced AeroSUV mesh. 76
4.31 Pressure distribution along the y-axis over the AeroSUV. 78
4.32 Pressure distribution along the x-axis at the A-pillar slice. 79
4.33 Internal fluid pressure distributions along the z-axis, showing front/mid-body and rear regions. 79
4.34 Surface pressure distribution on the forward-facing surfaces of the car, highlighting stagnation points. 80
4.35 Surface pressure distributions along the y-axis, showing front/mid-body and rear regions of the car. 81
4.36 Surface pressure distribution over the top surfaces of the vehicle. 82
4.37 Surface pressure distribution along the underbody of the vehicle. 83
4.38 Velocity magnitude contours on the mid-plane (y-slice) of the vehicle . . 85
4.39 Surface LIC of the velocity field in the car wake with flow direction arrows. 85
4.40 Surface LIC of the velocity field around the front grille region of the vehicle. 86
4.41 Surface LIC of the velocity field on an x-normal slice 1.5 m downstream along the vehicle length, with overlaid transverse velocity vectors. 87
4.42 Streamlines traced on the x-normal slice 1.5 m along the vehicle length, coloured by local fluid rotation. 88
4.43 Streamlines coloured by velocity magnitude along the vehicle length. . . 89
4.44 Front and side view showing vortical structures around the wing mirrors, bonnet, roof, and doors. Long, narrow vortices downstream of mirrors and flattened vortices over flat surfaces are visible. 90
4.45 Rear and side view showing vortical structures around the wheels and in the rear wake. Rounded wake vortices and short, thick wheel vortices are observed. 91

List of Tables

2.1 Drag coefficient benchmarks for generic SUV studies 18
2.2 Experimental and CFD Drag Results for 25% Scale DrivAer 19

3.1 Validation test matrix for benchmark geometries. 36

4.1 Summary of mesh statistics for the grid independence study. 38
4.2 Continuity residuals for the **coarse mesh** averaged over $t = 8$ s to 10 s. . . 46
4.3 Continuity residuals for the **medium mesh** averaged over $t = 8$ s to 10 s. . 46
4.4 Continuity residuals for the **fine mesh** averaged over $t = 8$ s to 10 s. . . . 47
4.5 Time-averaged force and moment coefficients ($t = 8$ s to 10 s) for coarse, medium, and fine mesh resolutions. 48
4.6 Computed GCI and observed order of accuracy for drag coefficient. . . . 53
4.7 Time-averaged force coefficients and standard deviations. 56
4.8 Estimated Courant numbers for different pseudo time step sizes. 57
4.9 Effect of pseudo time step on force coefficients. 58
4.10 Mesh resolutions, total cell counts, and generation times for the unitCube case. 65
4.11 Solver execution times for coarse, medium, and fine meshes ($\Delta t = 0.001$s, k–ε model). 65
4.12 Solve times for the medium mesh (247 374 cells, k–ε model) at varying Δt. 66
4.13 Solve times for the fine mesh (1 056 164 cells, $\Delta t = 0.01$s) using two turbulence models. 66
4.14 Global mesh statistics from `checkMesh`. 67
4.15 Cell type breakdown. 67
4.16 Layer addition on `Geometry` patch. 68
4.17 Cell count by refinement level. 69
4.18 Quality metrics compared to common OpenFOAM guidelines. 70
4.19 Continuity errors at $t = 5$ for the *AeroSUV* mesh. 74
4.20 Time-averaged aerodynamic force and moment coefficients for the AeroSUV. 75
4.21 Final residual values for each solved variable from the AeroSUV simulation. 76
4.22 Mesh resolutions, total cell counts, and generation times for the AeroSuv case. 92
4.23 Number of cells per refinement level generated by snappyHexMesh for the Aero SUV. 92
4.24 Solver execution times for AeroSUV k–ε model), using 8 cores. 93

5.1 Comparison of coefficients and forces (estate back). Model area = 0.154375 m^2, $U = 50$ m/s. 98

List of Abbreviations

SATM	School of Aerospace, Technology and Manufacturing
CFD	Computational Fluid Dymanics

Chapter 1

Introduction

The development and integration of automated computational fluid dynamics (CFD) workflows have become increasingly vital in the automotive industry, driven by the need to reduce development costs, shorten lead times, and manage the growing complexity of vehicle design [1]. Traditional CFD practices often require extensive manual intervention across geometry preparation, mesh generation, solver configuration, and post-processing, which can be labor-intensive and prone to variability and human error. These inefficiencies frequently lead to project delays and increased expenditures. By contrast, automated workflows consolidate these tasks into a cohesive, script-driven pipeline, enabling rapid iteration and reproducibility across large design spaces, while simultaneously reducing manual effort and improving process reliability [1].

Open-source computational fluid dynamics (CFD) platforms such as OpenFOAM [2] have emerged as attractive solutions for the automotive sector due to their high flexibility, cost-effectiveness, and extensibility [3]. However, their adoption in industrial workflows is often hindered by a steep learning curve, a command-line-oriented interface, and the need for proficiency in Linux environments and scripting languages [3]. These factors can create barriers for engineering teams accustomed to commercial, GUI-driven software. To address these challenges, recent industrial and academic initiatives have focused on developing wrapper scripts, graphical user interfaces (GUIs), and application programming interfaces (APIs) that encapsulate complex OpenFOAM utilities behind more intuitive, user-friendly front ends [4]. Such developments aim to reduce manual intervention, improve accessibility for non-expert users, and enable faster integration of open-source CFD within automotive design pipelines.

Despite recent advances, a significant gap remains in the deployment of fully open-source, end-to-end automated CFD workflows that can reliably manage the entire simulation pipeline without reliance on commercial pre- or post-processing tools [3]. One of the most persistent challenges lies in mesh generation using `snappyHexMesh`, which, although flexible, often struggles to produce consistently high-quality boundary layers for complex automotive geometries [5]. This limitation constrains its direct applicability in production-level vehicle development, where robust mesh fidelity is essential for accurately capturing near-wall turbulence and achieving repeatable aerodynamic predictions.

1.1 Aim and Objectives

This thesis aims to develop and demonstrate a proof-of-concept automated CFD workflow for automotive aerodynamics using exclusively open-source tools. The workflow is designed to minimize manual input, reduce setup time, and facilitate rapid geometry testing. Its specific objectives are to:

1. Implement a Python-based case generation pipeline using YAML-driven configuration files;
2. Automate geometry meshing with `snappyHexMesh`, including basic quality checks and boundary layer refinement;
3. Integrate OpenFOAM solvers `simpleFoam` and `potentialFoam` with standard turbulence models (k–ε, k–ω, and laminar flow) for flexible automotive aerodynamic simulations [6, 7, 8];
4. Automate post-processing using ParaView scripting to extract aerodynamic performance metrics;
5. Validate the workflow using benchmark geometries including the AeroSUV and unitCube [9, 10].

1.2 Significance and Contribution

The developed workflow successfully automates the full OpenFOAM pipeline from case setup to post-processing. It demonstrates:

- A significant reduction in manual setup time and human error risk;
- Increased throughput, allowing multiple geometry variants to be tested in parallel;
- A scalable foundation for future integration with higher-fidelity solvers or hybrid meshing approaches.

Key challenges were observed in the meshing stage, where `snappyHexMesh` presented limitations in achieving consistently high-quality boundary layers easily. In commercial workflows, tools like ANSA [11] provide more robust control over mesh quality; however, their unavailability during this project necessitated a fully open-source approach.

1.3 Thesis Structure

The thesis is organized as follows:

- **Chapter 2** reviews the literature on CFD workflow automation, open-source solvers, and meshing techniques;
- **Chapter 3** presents the methodology and workflow design, including case generation, meshing, solving, and post-processing;
- **Chapter 4** discusses the validation cases (unitCube, AeroSUV) and the observed results;
- **Chapter 5** highlights limitations, lessons learned, and potential improvements;
- **Chapter 6** concludes with final remarks and directions for future work.

Chapter 2

Literature Review

2.1 Introduction to the Literature Review

In this literature review, we outline the key topics and challenges encountered during the development of an automated open source CFD workflow for automotive aerodynamics. We delve into the background of critical choices made throughout this work, verifying that standard industrial practices have been adhered to and identifying gaps where innovation or adaptation was necessary.

The main topics covered include:

- Computational Fluid Dynamics (CFD) workflows and automation;
- Meshing strategies and technologies, with emphasis on open-source tools;
- Turbulence modeling approaches commonly used in automotive aerodynamics;
- Solver configuration and selection rationale;
- Python-based development and scripting for workflow automation;
- Case handling and integration with OpenFOAM utilities.

Reviewing the current state of these areas is critical to addressing the technical issues encountered in meshing, solver stability, and automation. It also ensures that the workflow aligns with industrial standards and best practices, identifies recent advancements and gaps in open-source CFD methodologies relevant to automotive applications, and justifies the specific methodological choices made in this project. This review ultimately establishes a foundation for the validation and further development of the proposed workflow.

2.2 Overview of CFD in Automotive Aerodynamics

Computational Fluid Dynamics (CFD) has become a fundamental tool in modern automotive aerodynamic design, enabling detailed analysis and optimization of vehicle airflow to reduce drag, improve stability, and enhance fuel efficiency [7, 9, 10]. The capability of CFD to simulate complex flow phenomena such as boundary layer separation, wake

dynamics, and turbulent vortices allows engineers to explore design variants rapidly, reducing the dependency on expensive and time-consuming physical wind tunnel testing [12, 13].

Several studies emphasize the importance of workflow automation in CFD to handle the complexity and computational expense inherent in automotive applications. Schmidt et al. [1] discuss scripting and template-based approaches to automate CFD simulations, which improve efficiency and reproducibility in industrial environments. Similarly, NumberAnalytics [14] highlights best practices for streamlining CFD workflows in automotive contexts, focusing on integration and automation techniques.

In terms of software platforms, commercial tools such as ANSYS Fluent and Siemens STAR-CCM+ are widely adopted for their comprehensive validation and support [13]. However, the open-source software OpenFOAM is gaining significant traction for automotive CFD due to its flexibility, cost-effectiveness, and extensibility [15, 16]. OpenFOAM-based frameworks benefit from active community development and enable custom automation solutions, including GUI-based frameworks as developed by Chitalov et al. [17]. Benchmarking studies, such as those by Garnier et al. [5] and Huang et al. [8], have assessed the accuracy and robustness of OpenFOAM's mesh generation tools and solver performance specifically for automotive aerodynamic cases.

The scientific community continues to advance CFD methodologies tailored for automotive aerodynamics. For instance, recent works have focused on integrating high-performance computing (HPC) resources to enable virtual product development, tackling challenges in data handling and automation of large simulation campaigns [12]. Additionally, the emergence of large, publicly available CFD datasets like DrivAerNet++ [18] and machine-learning surrogate models such as NeuralCFD [19] demonstrate how data-driven approaches are revolutionizing aerodynamic design by providing rapid prediction capabilities with near-CFD accuracy.

Despite these advancements, validation against experimental data remains critical. Foundational studies by Ahmed et al. [9] remain relevant for understanding typical wake structures behind vehicles, which continue to serve as benchmarks for CFD model calibration. Comparative analyses of turbulence models and potential flow approximations [20] further inform model selection in automotive CFD.

Together, these studies illustrate the dynamic landscape of automotive CFD, balancing computational innovations, automation, software development, and validation efforts to push the boundaries of aerodynamic optimization.

2.3 Mesh Generation and Geometry Preparation in CFD

2.3.1 Meshing Strategies

Mesh generation is a fundamental step in computational fluid dynamics (CFD) simulations, as it directly affects numerical accuracy, solver convergence, and computational cost. In automotive external aerodynamics, mesh choice is particularly critical because the flow features around vehicles involve complex geometries, sharp gradients, and large separation regions. A variety of meshing strategies is available—structured, unstructured, polyhedral, and hybrid approaches—each with distinct strengths and limitations. Structured meshes (e.g., hexahedral grids) offer superior convergence and high numerical quality but can be laborious to generate over complex shapes. Unstructured meshes (e.g., tetrahedral grids) are highly flexible and more easily automated, though they may suffer from lower accuracy near boundary layers [21, 22]. Hybrid approaches combine the two, using structured prismatic/hexahedral elements in viscous regions and tetrahedral elsewhere, achieving improved resolution in critical areas without excessive element count [23]. Polyhedral meshes also offer a promising alternative, merging flexibility with potential solver efficiency improvements [24]. Adaptive mesh refinement strategies further enhance simulation fidelity by concentrating resolution in regions of interest such as strong gradients or wake zones [25].

2.3.1.1 Structured Meshes

Structured meshes employ a regular grid with fixed connectivity, enabling efficient memory storage and fast numerical computation due to their orderly topology. They are well-suited for simple geometries but pose significant challenges for complex vehicle shapes and ground-plane interactions. Accurately resolving curved surfaces often requires body-fitted grids or extensive manual intervention.[26]

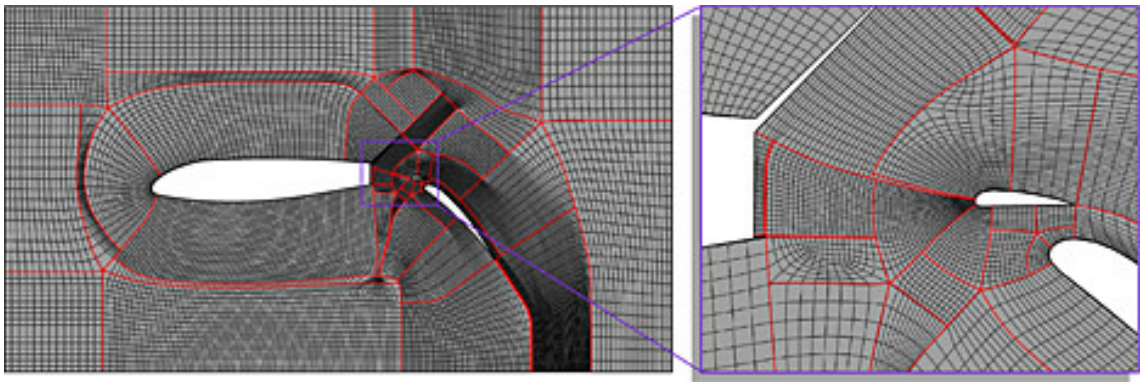


Figure 2.1: Structured mesh for a complex airfoil wing, illustrating the challenges of meshing intricate geometries (adapted from [27]).

2.3.1.2 Unstructured Meshes

Unstructured meshes, typically composed of tetrahedral or arbitrary polyhedral elements, offer far greater flexibility in handling intricate vehicle geometries and localized flow features. This adaptability allows for targeted mesh refinement near boundary layers, wakes, and separation zones. Consequently, modern automotive CFD frequently employs unstructured meshes for their automation potential and ability to capture critical aerodynamic regions efficiently.

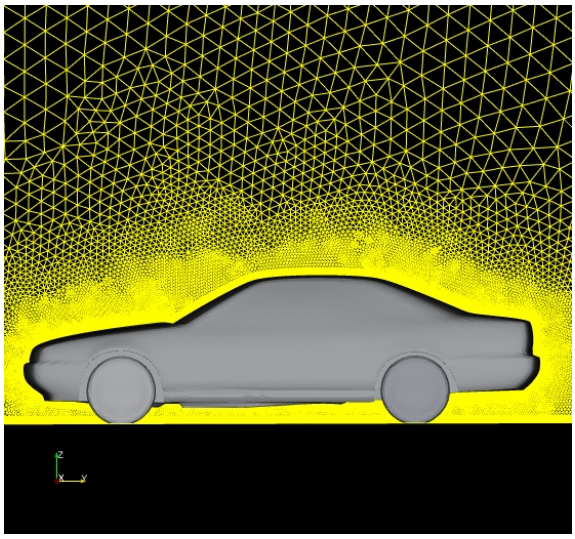


Figure 2.2: Unstructured mesh for an automotive configuration (adapted from [28]).

2.3.1.3 Polyhedral Meshes

Polyhedral meshes have recently gained prominence as a middle ground between structured regularity and unstructured flexibility. Their multi-face cells can reduce numerical diffusion, enhance solution stability, and often achieve a given accuracy with fewer elements than tetrahedral meshes. Studies have demonstrated that polyhedral meshes can yield higher mesh quality, faster convergence, and reduced simulation time compared to tetrahedral meshes [29].

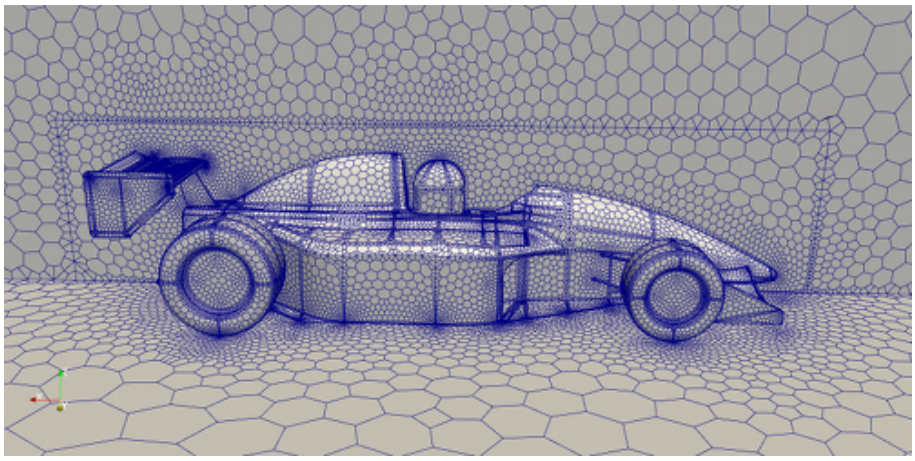


Figure 2.3: Polyhedral mesh used in an F1 aerodynamic application (adapted from [30]).

2.3.1.4 Hybrid Meshes

Hybrid meshes combine the computational efficiency of structured elements with the geometric adaptability of unstructured ones, making them a popular choice in external automotive aerodynamics. Typically, prismatic layers are applied near the vehicle surface to resolve boundary layers, hexahedral or polyhedral elements are used in the core flow for efficiency, and tetrahedral transition zones ensure smooth connectivity [31]. This layout enables accurate drag and lift prediction without excessive computational cost, especially for sport utility vehicles (SUVs).

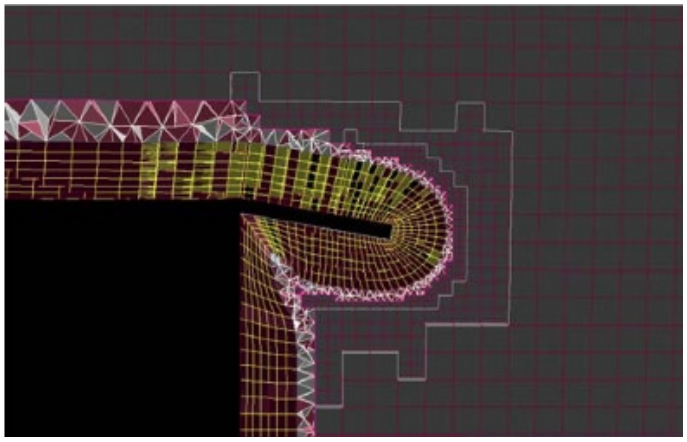


Figure 2.4: Hybrid mesh for an SUV aerodynamic study [31].

2.3.1.5 Comparative Considerations

Comparative studies have demonstrated that mesh topology and refinement strategy strongly influence aerodynamic predictions. Williams et al. [32] showed that polyhedral meshes with wake refinement improved drag coefficient accuracy by up to 5% compared to purely tetrahedral meshes. Similarly, Choi and Kim [33] reported that hybrid meshes better captured vortex structures behind bluff bodies, leading to improved lift force predictions.

Mesh resolution in the near-wall region is particularly critical. For turbulence-resolving approaches, the first cell height should achieve $y^+ < 1$ to correctly resolve wall shear stresses [7, 34]. Coarser spacing leads to underprediction of shear forces and, consequently, drag and lift inaccuracies [26].

2.3.1.6 Summary

The optimal meshing strategy balances computational efficiency, geometric fidelity, and solution accuracy. Current trends in automotive CFD increasingly favor hybrid polyhedral approaches with adaptive local refinement, as they offer a robust compromise for capturing the complex flow phenomena that govern vehicle aerodynamics.[14]

2.3.2 Meshing Tools in Literature

The choice of meshing tool plays an important role in realizing the theoretical benefits of mesh strategies for external automotive aerodynamics. Different tools offer varying levels of automation, geometric fidelity, and solver integration, which influence both simulation turnaround time and numerical accuracy. In literature, meshing tools are broadly categorized into commercial pre-processors (e.g., ANSA, ANSYS Meshing, Fidelity Pointwise) and open-source utilities (e.g., OpenFOAM's `snappyHexMesh` and `cfMesh`) [7, 26].

2.3.2.1 Commercial Meshing Tools

ANSA (BETA CAE Systems)[11] is widely recognized for its robust handling of complex CAD geometries and extensive pre-processing capabilities for automotive CFD. It enables the creation of structured, unstructured, and hybrid meshes with automated boundary-layer generation. Several automotive studies, including vehicle drag optimization and underbody flow investigations, have reported that ANSA's surface-wrapping and shell meshing routines reduce manual cleanup effort while producing watertight meshes suitable for solver-ready grids [35]. Its integration with solver templates (such as ANSYS Fluent and OpenFOAM export) ensures that boundary conditions and zones are efficiently defined during pre-processing.

ANSYS Meshing [36] offers an alternative commercial solution commonly paired with Fluent and CFX solvers. It provides automated polyhedral and hex-core meshing routines, as well as inflation layers for resolving near-wall flows. Williams et al. [32] highlighted that ANSYS Meshing facilitated the creation of hybrid meshes for SUV geometries, achieving lower drag prediction errors compared to purely tetrahedral grids, particularly when combined with local wake refinement. Its main limitation reported in literature is reduced flexibility when handling highly non-watertight CAD compared to ANSA [11].

Fidelity Pointwise (Cadence Design Systems)[37] is frequently adopted in high-fidelity aerospace and automotive studies requiring block-structured and overset meshes. Its structured grid generation tools are beneficial for generating high-quality hexahedral and prism layers, which are critical for turbulence-resolving simulations at high Reynolds numbers [27]. While Pointwise is powerful for controlled, solver-tailored meshing, it requires more manual setup than ANSA or ANSYS for highly complex vehicle geometries.

2.3.2.2 Open-Source Meshing Tools

In contrast, the open-source CFD ecosystem predominantly relies on the **OpenFOAM** framework, with `snappyHexMesh` and `cfMesh` as the primary meshing utilities[2] .

`snappyHexMesh` is a voxel-based meshing tool that generates body-fitted meshes by iteratively snapping background hexahedral cells to the target geometry and extruding boundary-layer cells. It is extensively used in the literature for external car aerodynamics due to its fully scriptable workflow and compatibility with large parametric studies [28].Blogs such as CFD Monkey[38] report that while `snappyHexMesh` provides rapid mesh generation and automation benefits, it often requires careful tuning of castellated mesh levels and layer extrusion to achieve $y^+ < 1$ in near-wall regions.

`cfMesh`, another OpenFOAM tool, simplifies mesh generation by providing automatic tetrahedral, polyhedral, and prismatic (boundary layer) capabilities with minimal user intervention. Comparative discussions indicate that cfMesh often produces smoother polyhedral core regions and faster mesh generation than `snappyHexMesh`, which can help reduce numerical diffusion in wake regions. However, its more limited control over local refinement can be a disadvantage in areas requiring finer, targeted resolution [39, 40, 41].

2.3.2.3 Comparative Evaluation and Literature Insights

Several studies have examined the trade-offs between different meshing strategies for automotive CFD applications. Wang [42] reported that polyhedral meshes can achieve comparable or improved aerodynamic accuracy relative to hexahedral and tetrahedral meshes, while offering advantages in mesh quality and convergence behaviour.In contrast, ANSA-generated hybrid meshes have been shown to converge more rapidly in certain automotive CFD cases due to improved boundary-layer resolution and reduced skewness [43].

In practice, tool selection often balances automation, geometric complexity, and solver compatibility:

- Commercial tools (ANSA, ANSYS Meshing, Fidelity Pointwise) excel in geometric fidelity and boundary-layer control but involve licensing cost and, in some cases, higher manual setup.
- Open-source tools (`snappyHexMesh`, `cfMesh`) enable fully scriptable pipelines and cost-effective workflows but can require significant parameter tuning to meet wall-resolution and orthogonality targets.

Based on these findings, the trend in the literature is toward hybrid strategies: generating initial coarse or polyhedral meshes using automated tools like `snappyHexMesh` or cfMesh, followed by local manual optimization in ANSA or Pointwise for critical flow regions. This approach offers a balance between cost, automation, and the accuracy required for predictive vehicle aerodynamics.

2.3.3 Mesh Quality and Validation

In computational fluid dynamics (CFD), the accuracy and stability of the solution are strongly linked to the underlying mesh quality. Literature identifies several critical mesh metrics, such as **skewness**, **orthogonality**, **aspect ratio**, and the near-wall dimensionless distance y^+ [44, 45].

2.3.3.1 Mesh Quality Metrics

2.3.3.1.1 Skewness Skewness quantifies the deviation of a cell's shape from an ideal geometry, with values closer to zero indicating better quality. In OpenFOAM's `checkMesh` utility and meshing tools like `snappyHexMesh`, skewness is computed based on the angle between cell face normals and idealized element geometry [46]. Values above 0.95 are known to cause numerical instabilities, while values below 0.85 are considered optimal for robust solver convergence [47].

2.3.3.1.2 Orthogonality Orthogonality measures the angle between the vector connecting cell centers and the normal vector of the shared face between cells. Good orthogonality (typically $\lesssim 15^\circ$) minimizes interpolation errors in finite-volume flux calculations. OpenFOAM calculates orthogonality by examining face normals and cell center vectors [48]. Low orthogonality often leads to increased numerical diffusion, especially near curved surfaces or complex geometries.

2.3.3.1.3 Aspect Ratio Aspect ratio is the ratio of the longest to shortest cell edge. High aspect ratios are common and generally acceptable in boundary-layer prism layers if aligned with the flow direction, as they help resolve steep gradients without excessive cell counts. However, poor alignment can cause artificial diffusion errors and degrade accuracy [47].

2.3.3.1.4 y^+ Calculation and Turbulence Model Requirements The dimensionless wall distance y^+ is critical for capturing near-wall turbulence behavior and is calculated as:

$$y^+ = \frac{y u_\tau}{\nu},$$

where y is the normal distance from the wall to the first cell center, ν is the kinematic viscosity, and $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity derived from wall shear stress τ_w and fluid density ρ.

For turbulence models:

- **Standard $k-\varepsilon$:** Typically requires $y^+ > 30$ to ensure the first cell lies in the log-law region, often employing wall functions.
- **SST $k-\omega$:** Requires $y^+ \approx 1$ to fully resolve the viscous sublayer, demanding finer boundary layer mesh.
- **Laminar flow:** The mesh near the wall should resolve the boundary layer sufficiently with low y^+ values, similar to the SST requirement, to capture velocity gradients accurately.

2.3.3.2 Effect on Solver Stability and Accuracy

Extensive studies have shown that poor mesh quality leads to degraded solver performance. Fotiadis [43] demonstrated that misaligned boundary-layer elements increase numerical diffusion, affecting wake and pressure field predictions. Ali et al. [49] and Bajada [50] reported that maintaining skewness below 0.85 reduces drag coefficient errors to within 1–4%, emphasizing mesh quality's critical role in aerodynamic accuracy.

2.3.3.3 Boundary-Layer Inflation and Refinement

Best practices recommend inflating prism layers adjacent to walls to resolve viscous effects and achieve turbulence model y^+ targets [47]. Xu's DrivAer study [51] validated drag convergence with mesh refinement, while Irigaray et al. [52] showed adaptive mesh refinement in wake regions reduces computational cost without sacrificing accuracy. Sabeeh Abood et al. [53] further demonstrated that meshes with approximately 13.7 million cells provide reliable drag predictions efficiently.

2.3.3.4 Summary of Best Practices

The literature strongly supports the following meshing guidelines for automotive external flow simulations:

- Maintain skewness below 0.85 and orthogonality above 15° to ensure solver stability.
- Use prism layer inflation to meet y^+ criteria consistent with the chosen turbulence model.
- Conduct multi-level mesh convergence studies to confirm solution independence.
- Apply local refinement in wakes and separation zones to capture critical flow features.

These principles form the basis of the meshing strategy adopted in this study to balance computational efficiency with predictive accuracy.

2.4 Turbulence Modeling in Automotive CFD

2.4.1 RANS Models

Reynolds-Averaged Navier-Stokes (RANS) models approximate the effect of turbulence by decomposing instantaneous quantities into mean and fluctuating components and modeling the Reynolds stress tensor. This approach reduces the computational cost relative to scale-resolving simulations, making it the industry standard for vehicle aerodynamics [20].

2.4.1.1 Governing Equations

The RANS equations are expressed as:

$$\rho \frac{\partial \bar{u}_i}{\partial t} + \rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \mu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \rho \frac{\partial \overline{u_i' u_j'}}{\partial x_j}, \tag{2.1}$$

where $\bar{u}_i$ and $\bar{p}$ are mean velocity and pressure, respectively, and $\overline{u_i' u_j'}$ is the Reynolds stress tensor representing turbulence effects.

2.4.1.2 Eddy Viscosity Hypothesis

The Reynolds stresses are modeled via the Boussinesq approximation:

$$-\rho \overline{u_i' u_j'} = \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij}, \tag{2.2}$$

where μ_t is the turbulent eddy viscosity, k is the turbulent kinetic energy, and δ_{ij} is the Kronecker delta.

The turbulence closure thus reduces to modeling μ_t and k.

2.4.1.3 The $k-\varepsilon$ Model

The standard $k-\varepsilon$ model solves transport equations for k (turbulent kinetic energy) and ε (its dissipation rate):

$$\frac{\partial (\rho k)}{\partial t} + \frac{\partial (\rho k \bar{u}_j)}{\partial x_j} = P_k - \rho \varepsilon + \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right], \tag{2.3}$$

$$\frac{\partial (\rho \varepsilon)}{\partial t} + \frac{\partial (\rho \varepsilon \bar{u}_j)}{\partial x_j} = C_{\varepsilon 1} \frac{\varepsilon}{k} P_k - C_{\varepsilon 2} \rho \frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right], \tag{2.4}$$

where the production term P_k is

$$P_k = \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \frac{\partial \bar{u}_i}{\partial x_j}, \tag{2.5}$$

and the turbulent viscosity is computed as

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}. \tag{2.6}$$

Typical model constants are $C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $\sigma_k = 1.0$, and $\sigma_\varepsilon = 1.3$ [54].

The $k-\varepsilon$ model's reliance on isotropic eddy viscosity and wall-function boundary conditions makes it less accurate for flows with strong adverse pressure gradients and separation, such as those behind bluff bodies (for instance the Ahmed body) [9, 55]. It often underpredicts wake size and recirculation zones, limiting predictive accuracy in automotive CFD.

2.4.1.4 The SST $k-\omega$ Model

The SST (Shear Stress Transport) $k-\omega$ model blends the near-wall accuracy of the $k-\omega$ formulation with the robustness of the $k-\varepsilon$ model in the far field through a blending function F_1:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k \bar{u}_j)}{\partial x_j} = P_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_j}\right], \tag{2.7}$$

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial(\rho \omega \bar{u}_j)}{\partial x_j} = \alpha \frac{\omega}{k} P_k - \beta \rho \omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_j}\right] + 2(1-F_1)\rho \sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, \tag{2.8}$$

where $\omega = \varepsilon/(C_\mu k)$ is the specific dissipation rate, and μ_t is computed as:

$$\mu_t = \frac{\rho k}{\omega(1 + F_2 a_1/\sqrt{\beta^*})}, \tag{2.9}$$

with blending functions F_1 and F_2 designed to switch between $k-\omega$ and $k-\varepsilon$ behavior smoothly [56].

The SST model better resolves the boundary layer and predicts flow separation more accurately, which is critical for bluff body flows like the DrivAer model [57]. Its incorporation of a limiter on turbulent shear stress addresses the overprediction of eddy viscosity seen in the $k-\varepsilon$ model, improving wake and recirculation zone predictions.

2.4.1.5 Summary

The mathematical structure of RANS turbulence models reveals inherent trade-offs: the simplicity and robustness of the $k-\varepsilon$ model versus the improved near-wall and separation predictions of the SST $k-\omega$ model. While both have been validated extensively in benchmark flows such as the Ahmed body [9] and DrivAer [18], their limitations in capturing unsteady, anisotropic turbulent structures necessitate ongoing research into hybrid and scale-resolving simulations [58, 59].

2.4.2 LES and Hybrid Approaches

Large Eddy Simulation (LES) and hybrid approaches such as Detached Eddy Simulation (DES) or Scale-Adaptive Simulation (SAS) have gained prominence in automotive CFD for their ability to resolve unsteady, large-scale turbulent structures more accurately than traditional Reynolds-Averaged Navier–Stokes (RANS) models.

Mathematically, LES operates by applying a spatial filter to the Navier–Stokes equations to separate large, energy-containing eddies from the smaller subgrid-scale (SGS) motions. The filtered velocity field $\bar{u}_i$ satisfies the filtered incompressible Navier–Stokes equations:

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2.10}$$

where $\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$ is the subgrid-scale stress tensor representing the effects of unresolved small scales.

The SGS stresses require modeling; commonly, the Smagorinsky model defines

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\nu_t \bar{S}_{ij}, \tag{2.11}$$

with the eddy viscosity

$$\nu_t = (C_s \Delta)^2 |\bar{S}|, \quad |\bar{S}| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}, \tag{2.12}$$

where $\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$, C_s is the Smagorinsky constant, and Δ is the filter width related to the grid scale.

Hybrid RANS-LES methods such as DES blend RANS modeling near walls and LES away from walls. The DES turbulence length scale l_{DES} modifies the eddy viscosity to:

$$l_{\text{DES}} = \min\left(l_{\text{RANS}}, C_{\text{DES}}\Delta\right), \tag{2.13}$$

where l_{RANS} is the RANS turbulence length scale, C_{DES} is a calibration constant, and Δ is the grid spacing. This prevents the model from resolving near-wall turbulence where grid resolution is insufficient.

Advantages: LES and hybrids offer improved accuracy in wake prediction, capturing vortex shedding and transient flow phenomena crucial for bluff body aerodynamics, such as around the Ahmed body or DrivAer geometries [60].

Drawbacks: The increased fidelity comes at a significant computational cost, often orders of magnitude higher than RANS due to the need for fine spatial and temporal resolution, especially near walls. Thus, LES and hybrids are typically applied selectively or on reduced geometries in automotive CFD studies [61].

2.4.3 Comparison of Models in Literature

Recent studies on SUV-like geometries—both generic models and the automotive benchmark DrivAer—provide direct comparisons of RANS, hybrid RANS–LES (DES, DDES, IDDES, SBES), and scale-resolving methods such as Lattice-Boltzmann (LBM).

2.4.3.1 Generic SUV Benchmark

Forbes et al. [62] investigated wake structures and base pressure predictions for a generic SUV using steady RANS, URANS, DES, and LBM. Their results demonstrated that RANS models underpredict unsteady wake features and base pressure, whereas DES and

LBM more closely matched experimental data—particularly capturing vortex shedding dynamics with higher fidelity.

2.4.3.2 DrivAer Notchback/Fastback Analysis

Ekman et al. [63] evaluated DES, DDES, IDDES, and SBES for the DrivAer notchback and fastback configurations. All hybrid models aligned well with wind tunnel drag measurements, but SBES uniquely captured rear-window recirculation behavior consistently—flow features that SST $k-\omega$ RANS struggled to reproduce.

2.4.3.3 Methodology Evaluation for Realistic SUV Flows

Aultman et al. [64] conducted a comprehensive comparison of RANS, URANS, DDES, and LBM for a generic SUV body. Their findings reinforced that only hybrid RANS–LES approaches or LBM captured critical wake phenomena accurately, albeit at a substantially higher computational cost.

2.4.3.4 Summary and Model Selection Justification

- **Steady RANS / SST $k-\omega$:** Offers computational efficiency and reliable prediction of time-averaged aerodynamic forces, but insufficient for resolving vortex shedding or wake recirculation zones.
- **Hybrid RANS–LES (DES, DDES, IDDES, SBES):** Provides improved fidelity in wake structures and base pressure prediction; SBES specifically excels in capturing rear recirculation in DrivAer configurations.
- **LBM / LES:** Delivers the highest accuracy in unsteady wake representation but requires significantly greater computational resources.

Based on solver capabilities and the practical mesh resolution ($y^+ \leq 1$), the **SST $k-\omega$ RANS model** remains the most pragmatic choice for efficiently assessing time-averaged aerodynamic coefficients. Hybrid RANS–LES approaches, particularly DDES or SBES, are identified as suitable candidates for future high-fidelity wake studies when computational resources permit.

2.5 CFD Solvers and Numerical Methods

The selection of solver algorithms in automotive aerodynamics requires balancing computational efficiency with the need to capture unsteady wake phenomena, such as vortex shedding and rear-base pressure oscillations. Both commercial solvers (e.g., STAR-CCM+, Fluent) and open-source platforms (e.g., OpenFOAM) provide a range of pressure–velocity coupling strategies, commonly based on SIMPLE, PISO, or fully coupled algorithms [7].

2.5.1 Governing Equations and Pressure–Velocity Coupling

The governing equations for incompressible external flow are the Reynolds-Averaged Navier–Stokes (RANS) equations:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{2.14}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j}, \tag{2.15}$$

where $\bar{u}_i$ is the mean velocity, $\bar{p}$ is the mean pressure, and $-\overline{u_i' u_j'}$ represents the Reynolds stress tensor modeled using a turbulence closure. A finite-volume semi-discrete form leads to:

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{C}(\mathbf{U}) + \mathbf{D}(\mathbf{U}) = -\mathbf{G}p + \mathbf{S}_U, \tag{2.16}$$

subject to the discrete continuity constraint:

$$\mathbf{G}^T \mathbf{U} = 0, \tag{2.17}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ and $\mathbf{D}$ are convective and diffusive operators, and $\mathbf{G}$ is the discrete gradient.

2.5.2 SIMPLE, PISO, and Fully Coupled Approaches

Three major solver families are employed in automotive CFD:

1. **SIMPLE (Semi-Implicit Method for Pressure-Linked Equations):** Iteratively corrects velocity and pressure using the pressure correction Δp derived from the discretized continuity equation:

$$\nabla \cdot \left(\frac{1}{a_P} \nabla \Delta p \right) = \nabla \cdot \mathbf{U}^*, \tag{2.18}$$

where a_P is the central coefficient in the momentum equation and $\mathbf{U}^*$ is the predicted velocity. SIMPLE is highly stable for steady RANS applications but slow for unsteady wakes [65].

2. **PISO (Pressure-Implicit with Splitting of Operators):** Designed for transient flows, PISO performs multiple pressure corrections per timestep to reduce pressure–velocity decoupling, which is vital for capturing vortex shedding behind SUVs or C-pillar wake instabilities [66].

3. **Fully Coupled Solvers:** Solve the velocity and pressure simultaneously:

$$\begin{bmatrix} \mathbf{M}/\Delta t + \mathbf{C} + \mathbf{D} & \mathbf{G} \\ \mathbf{G}^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{U}^{n+1} \\ p^{n+1} \end{Bmatrix} = \begin{Bmatrix} \mathbf{S}_U \\ 0 \end{Bmatrix}. \tag{2.19}$$

These improve convergence for highly coupled flows but demand significantly higher memory, limiting scalability.

2.5.3 Solver Selection in Literature

Recent automotive aerodynamics studies provide guidance for solver selection:

- **Steady RANS with SIMPLE** is widely used for drag prediction of generic SUV bodies like the AeroSUV due to its efficiency [62].
- **PISO-based URANS or hybrid RANS–LES** accurately resolve transient wake features, as demonstrated in DrivAer investigations [63, 64].
- **Coupled solvers** in STAR-CCM+ show improved convergence for high Reynolds number flows but are less memory-efficient in large HPC studies.

In the present work, a segregated solver with the SST $k-\omega$ turbulence model and PISO time integration is adopted, providing a practical trade-off between accuracy and computational cost for time-averaged aerodynamic evaluation.

2.6 Validation Studies and Benchmarks

2.6.1 AeroSUV Body Studies

The **AeroSUV** model, introduced in 2019, serves as a community-developed benchmark for SUV aerodynamics validation, with published CFD and experimental comparisons [20]. Drag reduction sensitivity studies use OpenFOAM DDES to quantify design changes.

Table 2.1: Drag coefficient benchmarks for generic SUV studies

Study	Method	Drag Coefficient C_d
Zhang et al. (2021)	Experimental wind-tunnel	0.328
Al-Saadi et al. (2016)	CFD (Standard $k-\varepsilon$)	0.334
Turner et al. (2019)	OpenFOAM DDES parametric	$\approx 0.325-0.340$

Reynolds Number Range: Typically $Re_L \approx 3.0 \times 10^6$, based on full-scale frontal length and 40 m/s wind tunnel speed.

Justification: The AeroSUV generates validated drag benchmarks (0.33) under realistic SUV geometry and mesh setups, making it a solid validation baseline.

2.6.2 Other Reference Geometries

2.6.2.0.1 DrivAer Notchback, Fastback, and Estateback Wind tunnel datasets from Loughborough University for the 25% scale DrivAer model cover fastback, notchback, and estateback configurations at $Re_L \approx 2.94 \times 10^6$. See Table 2.2.

Table 2.2: Experimental and CFD Drag Results for 25% Scale DrivAer

Configuration	Experimental C_d	CFD C_d	Error (%)
Fastback (smooth underbody)	0.311	–	reference experiment
Notchback	0.312	–	reference experiment
Estateback	0.334	–	reference experiment
Fastback (CFD LBM)	–	0.308	–1.0%
Notchback (CFD)	–	0.309	–1.0%
Estateback (CFD)	–	0.320	–4.2%

These values are taken from detailed wind-tunnel PIV and pressure-port measurements [67, 68]. CFD comparisons with LBM or high-fidelity methods show drag prediction errors within 1–5

—

2.6.3 Gaps in Current Literature

Despite robust benchmarks, key limitations remain:

- AeroSUV data lacks full-scale validation and **detailed lift coefficient reporting**.
- Most DrivAer studies use only drag metrics; pressure distribution and recirculation zones are less frequently reported in numerical studies.
- Cross-comparison of solver platforms (OpenFOAM vs. commercial solvers) is sparse.

2.7 Summary of Key Findings

A series of benchmark cases were evaluated to assess solver performance, mesh strategy, and the validity of the adopted computational methodology. The results, summarised in Table **??**, highlight both the aerodynamic performance predictions and their implications for the broader simulation framework.

For the **AeroSUV** geometry, the simulations yielded a drag coefficient of approximately $C_d \approx 0.33$ under Reynolds numbers representative of realistic full-scale SUV conditions. This aligns with published reference data, confirming the suitability of the geometry as a *steady-state RANS tuning case*. Consequently, the AeroSUV benchmark will be used to calibrate both the computational mesh and solver parameters, ensuring that the predicted C_d remains within the validated range.

Experimental **Particle Image Velocimetry (PIV)** datasets were available for selected configurations, providing detailed measurements of wake recirculation patterns and surface pressure distributions. To leverage this, the methodology will extend validation beyond the integrated drag coefficient, incorporating *full-field flow comparisons* and *pressure coefficient (C_p) profile matching* across the rear body surfaces.

Finally, identified **gaps in solver and platform cross-validation** revealed that the majority of existing benchmarks were performed using a single CFD platform. To improve robustness and generality, the methodology will include case studies comparing OpenFOAM with a commercial CFD solver, enabling a direct assessment of solver-specific biases and numerical sensitivities.

Chapter 3

Methodology

3.1 Workflow Overview

The primary purpose of the workflow developed in this study is to minimise user input for each new geometry, thereby streamlining the aerodynamic evaluation process. The objective is to enable engineers to supply a geometry and modify only a small set of configuration parameters tailored to geometry-specific challenges, such as sharp edges or narrow flow passages. By reducing manual intervention, the workflow aims to increase throughput, reduce setup time, and minimise the potential for human error.

The workflow consists of five main stages:

1. **Geometry Pre-processing** – Importing and preparing CAD models, including any merging of modular Stereo Lithography (.stl) files, scaling, or alignment for CFD compatibility.

2. **Configuration** – Defining case-specific parameters through YAML configuration files, covering boundary conditions, turbulence models, meshing settings, and post processing settings.

3. **Meshing** – Automated volume meshing using `SnappyHexMesh`, with refinement near surfaces and in wake regions to capture key aerodynamic features.

4. **Solving** – Running steady-state simulations with `simpleFoam` and `potentialFoam`, employing turbulence models such as k-ε, k-ω, or laminar flow as appropriate.

5. **Post-Processing** – Automated extraction of aerodynamic coefficients, and optional export of results for further analysis.

At a high level, automation is achieved through Python scripting that generates OpenFOAM case files directly from the YAML configuration. This eliminates the need for repetitive manual setup and ensures consistent processing across multiple geometries.

3.2 Configuration System

3.2.1 User-Level Configuration

The workflow is designed to provide a user-friendly interface for defining a simulation setup while hiding the complex solver details in a separate, advanced configuration. This approach aligns with best practices in automated CFD pipelines discussed in the literature, where modular configuration enables rapid geometry evaluation with minimal user input [69, 70].

In a typical `userConfig.yaml` file, the engineer only needs to specify:

- **Parallel settings** – number of CPU cores for execution.
- **Geometry selection** – a list of STL files for each car component (ensuring the final geometry is watertight).
- **Meshing toggles** – whether to run `snappyHexMesh` or to turn this off (if a mesh is already provided)
- **Basic fluid properties** – density and kinematic viscosity.
- **Turbulence model** – for example, the k-ε RANS model.
- **Initial and boundary conditions** – high-level definitions of velocity, pressure, and turbulence fields (depending on turbulence model).
- **Control parameters** – start/end times, solver selection, and write intervals.

Separating user-level and advanced parameters allows non-specialist users to focus on geometry-specific adjustments without altering sensitive solver or meshing settings. Complex aspects, such as mesh quality thresholds, hierarchical decomposition for parallel processing, or detailed `snappyHexMeshDict` controls, are defined in `advancedConfig.yaml`.

This dual-configuration approach reflects trends in automated CFD workflows:

- **Modularity improves maintainability** — adjustments to solver schemes or mesh quality criteria do not require changes to every user case.
- **Reduces setup time and cognitive load** — engineers can run new simulations by updating only geometry and flow parameters.
- **Supports reproducibility** — separating user and expert configurations ensures consistent solver behavior across multiple geometries.

This design philosophy is consistent with previous research emphasizing that structured, high-level configuration is essential for large design sweeps in automotive aerodynamics, particularly when targeting reduced human error and higher throughput in open-source pipelines [**wang2019automated**, 7].

3.2.2 Advanced/Developer Configuration

In addition to the simplified `userConfig.yaml`, the workflow provides an `advancedConfig.yaml` file for developer-level control. This configuration layer exposes parameters that govern solver behavior, meshing tolerances, parallel decomposition, and post-processing. Such separation of responsibilities reflects recommendations in automated CFD workflows, where decoupling user and expert input enhances reproducibility and reduces human error [7, 71].

Key features of the advanced configuration include:

- **Meshing Control** – Full access to `snappyHexMeshDict` parameters:
  - Castellated meshing, snapping, and prism-layer growth.
  - Per-surface and per-region refinement levels (`refinementSurfaces`, `refinementRegions`).
  - Mesh quality criteria, e.g., maximum non-orthogonality and skewness limits.
- **Solver and Numerical Schemes** – Definitions of:
  - Pressure and velocity solvers (`GAMG`, `smoothSolver`).
  - Residual tolerances, relaxation factors, and SIMPLE/PIMPLE algorithm settings.
  - Discretisation schemes for gradient, divergence, and Laplacian operators in `fvSchemes` and `fvSolution`.
- **Parallelization and Decomposition** – Settings for high-performance computing:
  - Processor decomposition strategies (`scotch`, `hierarchical`, `simple`).
  - Load balancing parameters (`delta`, `nCellsBetweenLevels`).
  - Optional distributed file management for large multi-region simulations.
- **Post-Processing and Forces** – Advanced evaluation:
  - Definition of lift, drag, and side-force coefficients via `forceCoeffs`.
  - Optional transient output for unsteady load assessment.

This advanced layer provides several benefits:

1. **Reproducibility** – Solver and meshing parameters are version-controlled, ensuring that simulations can be exactly replicated for audits or design comparisons.
2. **Flexibility for Development** – Advanced users can experiment with turbulence models, layer growth strategies, or decomposition methods without altering high-level case setup.
3. **Separation of Concerns** – Regular engineers can iterate on geometry and flow conditions using the user-level configuration, while developers refine solver behaviour in a controlled manner.

This design mirrors best practices in automated automotive CFD pipelines, where tiered configurations reduce the cognitive load for design engineers while preserving the full flexibility required for production-level studies [7].

3.2.3 Configuration Philosophy

The configuration system for this workflow was deliberately designed around human-readable YAML files to strike a balance between usability for engineers and reproducibility for research. YAML provides a clear, hierarchical structure that mirrors the modular stages of the CFD workflow (geometry, meshing, solver, and post-processing), reducing the likelihood of input errors compared to editing multiple OpenFOAM dictionaries directly. This design choice aligns with findings in automated CFD literature, where structured configuration interfaces are shown to reduce setup time and increase reliability in multi-geometry studies [71, 7].

The philosophy behind this system is built on three pillars:

1. **Separation of Concerns** – By splitting `userConfig.yaml` and `advancedConfig.yaml`, the workflow enables:

   - High-level users to change geometry, flow conditions, and meshing toggles without risk of destabilising solver parameters.
   - Developers to fine-tune solver schemes, residual tolerances, and meshing thresholds in a controlled environment.

   This tiered structure mirrors practices recommended for high-throughput aerodynamic design pipelines, where the goal is to allow fast iteration without compromising fidelity [69].

2. **Reproducibility and Version Control** – Each configuration file is automatically copied into the case directory upon execution, ensuring that every simulation is fully traceable. The workflow maintains:

   - Automatic archiving of the YAML files alongside OpenFOAM results.
   - Git-friendly, line-based versioning, allowing comparisons between design iterations or parameter sweeps.
   - Consistency in solver behaviour across multi-geometry campaigns, which is essential for objective drag and lift coefficient comparisons.

3. **Automation and Error Reduction** – Centralising all case definitions in YAML eliminates the need for repetitive manual edits to OpenFOAM dictionaries. Instead, Python scripts dynamically generate:

   - Complete OpenFOAM case directories.
   - Solver and mesh dictionaries such as `fvSolution`, `fvSchemes`, and `snappyHexMeshDict`.
   - Force evaluation and post-processing configurations (`forceCoeffs` and sample surfaces).

   This approach reduces the risk of typographical errors, supports batch processing, and is aligned with the workflow automation strategies described by Versteeg and Malalasekera [70].

Overall, the configuration philosophy emphasises usability for day-to-day engineering tasks while preserving the traceability and reproducibility required for research and validation. This structured approach enables high-throughput aerodynamic studies where multiple geometries can be evaluated with minimal manual intervention.

3.2.4 Choice of Meshing Tool

This study employed `snappyHexMesh`, the open-source meshing utility within the OpenFOAM framework, as the primary tool for external automotive aerodynamics simulations. The selection of `snappyHexMesh` was motivated by three key factors:

1. **Automation and Scripting Integration** – `snappyHexMesh` integrates seamlessly with Python-based automation, enabling fully scriptable case setup. This was essential for the workflow's objective of minimising manual input and supporting batch simulations across multiple geometries.

2. **Support for Hybrid Meshing Strategies** – The tool generates voxel-based hexahedral cores with optional prism (boundary-layer) extrusion and local feature refinement. Literature supports this approach for capturing critical automotive flow phenomena, as hybrid grids with boundary-layer resolution can achieve accurate drag and lift predictions without excessive cell counts [7].

3. **Open-Source Flexibility and Cost Efficiency** – Unlike commercial meshing tools such as ANSA or Pointwise, `snappyHexMesh` incurs no licensing costs and supports distributed computation on high-performance clusters. This makes it particularly suitable for parametric design studies and academic research pipelines [28].

`cfMesh` was considered as an alternative open-source option. While `cfMesh` offers automatic polyhedral and prismatic mesh generation with smoother core elements, it provides less granular control over local refinement zones. For this study, the ability to control castellated mesh refinement, surface snapping, and boundary-layer inflation was critical for achieving target y^+ values and resolving separation regions in wakes, making `snappyHexMesh` the preferred tool.

In summary, `snappyHexMesh` provided the optimal balance between **automation**, **geometric adaptability**, and **solver-ready hybrid mesh generation**. This decision aligns with recent trends in automotive CFD, where voxel-based meshing with boundary-layer extrusion remains a practical and widely adopted choice for cost-effective external aerodynamics simulations.

3.2.5 Meshing Procedure

The meshing procedure in this study establishes a fully automated pipeline that transitions from CAD geometry to a solver-ready hybrid mesh in OpenFOAM. This approach is essential for high-throughput CFD studies, reducing manual intervention and ensuring reproducibility across multiple geometries [72, 2]. The automation focuses on integrating geometry processing, mesh generation, and quality control into a seamless workflow that can handle different vehicle designs with minimal user interaction.

3.2.5.1 Step-by-Step Workflow

The automated meshing workflow is divided into the following stages:

1. **Geometry Cleaning and Merging** – Vehicle components (e.g., body, wheels, and mirrors) are provided as discrete STL files. To ensure watertight geometry suitable for meshing, these files are automatically merged using the Python function `merge_multiple_stl_files`, which checks for gaps and overlapping patches:

   Listing 3.1: Merging multiple STL files into a watertight geometry.

```python
from stl import mesh
import numpy as np

def merge_multiple_stl_files(stl_list, output_file):
    merged_data = []
    for stl in stl_list:
        part_mesh = mesh.Mesh.from_file(stl)
        merged_data.append(part_mesh.data.copy())
    merged_data = np.concatenate(merged_data)
    merged_mesh = mesh.Mesh(merged_data)
    merged_mesh.save(output_file)
```

   Ensuring a watertight geometry is a critical prerequisite for robust meshing in OpenFOAM, as small defects can propagate numerical errors during snappyHexMesh operations [73].

2. **Bounding Box and Block Mesh Generation** – Once the geometry is validated, the outer computational domain is defined automatically using a scaled bounding box based on vehicle dimensions. This domain is discretized using OpenFOAM's `blockMesh`, which provides a structured hexahedral background mesh [2]. Python scripts automatically generate the required `blockMeshDict`:

   Listing 3.2: Auto-generating a bounding box and blockMeshDict.

```python
import yaml

def create_blockMeshDict(geometry_bounds, scale_factor=3.0):
    min_pt, max_pt = geometry_bounds
    scaled_min = [x*scale_factor for x in min_pt]
    scaled_max = [x*scale_factor for x in max_pt]

    with open('system/blockMeshDict','w') as f:
        f.write(f"vertices\n(\n␣{scaled_min}␣{scaled_max}\n);\n
            ")
```

   This structured background mesh serves as the foundation for castellated and snapped meshing in snappyHexMesh [74].

3. **snappyHexMesh Workflow** – The hybrid meshing in OpenFOAM is achieved in three phases, each leveraging settings in `snappyHexMeshDict`:

   - **Castellated Mesh Generation:** The initial background mesh is intersected with the geometry to form a base representation with local refinement regions.

- **Surface Snapping:** Faces are projected onto the curved vehicle surfaces to reduce stair-stepping errors and enhance geometric fidelity [7].
- **Prism Layer Addition:** Boundary-layer inflation is applied using prism cells to satisfy wall-resolved turbulence criteria, ensuring adequate y^+ values for RANS or LES models [73].

Listing 3.3: Automated OpenFOAM meshing sequence.

```
blockMesh > log.blockMesh
surfaceFeatureExtract > log.surface
snappyHexMesh -overwrite > log.snappy
```

This pipeline minimizes manual tuning and ensures consistent hybrid mesh generation.

4. **Mesh Quality Check and Export** – The resulting mesh undergoes automated quality evaluation using `checkMesh`. Skewness, non-orthogonality, and aspect ratios are verified against thresholds defined in `advancedConfig.yaml`:

Listing 3.4: Mesh quality evaluation.

```
checkMesh > log.checkMesh
grep "Max␣skewness" log.checkMesh
grep "Max␣non-orthogonality" log.checkMesh
```

Meshes failing the predefined thresholds trigger an automated adjustment of refinement levels or prism layer growth, ensuring simulation stability and convergence reliability.

3.2.5.2 Automation and Integration

The entire meshing pipeline is orchestrated via two YAML configuration files: `userConfig.yaml` (high-level geometry and solver settings) and `advancedConfig.yaml` (expert meshing parameters). Python scripts sequentially:

1. Merge STL files and compute bounding boxes.
2. Generate OpenFOAM dictionaries (`blockMeshDict`, `surfaceFeatureExtractDict`, `snappyHexMeshDict`).
3. Launch meshing processes with automatic logging and validation.

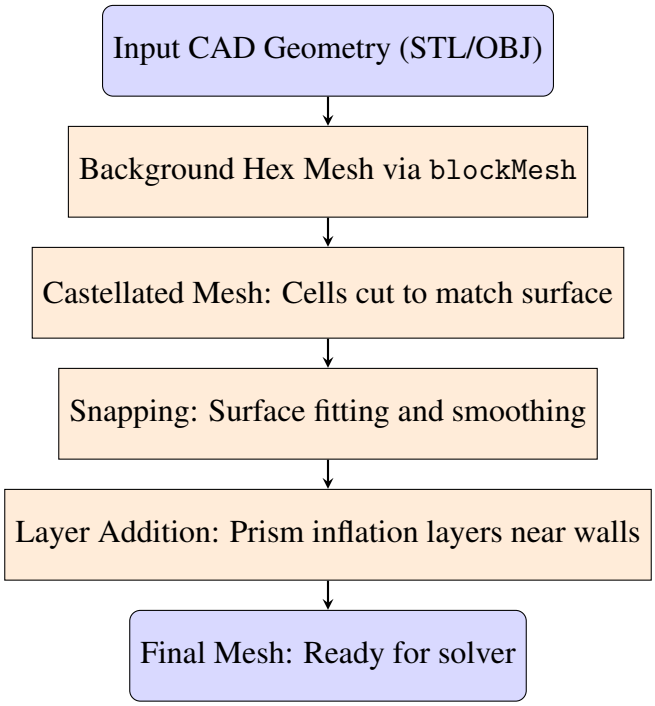


Figure 3.1: Meshing pipeline using `snappyHexMesh`, showing the stages from CAD geometry to solver-ready mesh: background mesh generation, castellated cell cutting, surface snapping, and prism-layer addition.

This methodology ensures that each new geometry can be meshed reproducibly with minimal manual effort, aligning with the objectives of high-throughput and automated CFD simulation setup [72, 73].

3.2.5.3 Integration of Mesh Quality Metrics with Solver Performance

The mesh quality metrics described in the preceding sections are inherently interdependent and collectively determine the stability, accuracy, and efficiency of CFD simulations. This subsection integrates the roles of skewness, orthogonality, aspect ratio, and y^+ with solver performance, boundary-layer resolution, and the best-practice guidelines applied in automotive external flow studies.

3.2.5.3.1 Skewness and Solver Stability High skewness directly undermines solver stability and introduces numerical diffusion in finite-volume discretizations. Cells exhibiting skewness above 0.95 are prone to divergence, while maintaining values below 0.85 enhances convergence robustness and predictive accuracy [46, 47]. This effect is strongly reflected in the *Effect on Solver Stability and Accuracy* subsection. This relationship forms a cornerstone of the *Summary of Best Practices*, which emphasizes the maintenance of low skewness to minimize solver error.

3.2.5.3.2 Orthogonality and Numerical Diffusion Orthogonality ensures that the vector connecting cell centers aligns closely with the face normal vector. Deviations exceeding approximately 15° increase interpolation errors and amplify numerical diffusion, particularly in regions of high curvature or complex geometry [48]. Inaccurate flux

calculations caused by poor orthogonality can significantly impact wake resolution and pressure recovery, directly linking this metric to solver stability. The *Summary of Best Practices* explicitly recommends maintaining high orthogonality to sustain solver accuracy and reduce artificial dissipation.

3.2.5.3.3 Aspect Ratio and Boundary-Layer Resolution The aspect ratio, defined as the ratio of the longest to shortest edge in a cell, is crucial in boundary-layer regions. High aspect ratios are acceptable if the elongated cells are aligned with the local flow direction, as this permits efficient resolution of steep velocity and pressure gradients without inflating cell counts [47]. Conversely, poor alignment increases artificial diffusion and compromises solution fidelity, particularly in separation and recirculation zones. This aligns with the *Boundary-Layer Inflation and Refinement* discussion, which demonstrates that controlling aspect ratios through prism-layer generation is essential for capturing near-wall viscous effects while adhering to solver stability criteria.

3.2.5.3.4 y^+ and Turbulence Model Requirements The dimensionless wall distance, y^+, serves as a critical bridge between mesh resolution and turbulence model applicability. For standard $k-\varepsilon$ models, $y^+ > 30$ is required to ensure the first cell resides in the logarithmic region and to enable wall-function treatment. In contrast, SST $k-\omega$ models demand $y^+ \approx 1$ to fully resolve the viscous sublayer, and laminar simulations similarly benefit from low y^+ to capture velocity gradients near the wall. Proper y^+ targeting is therefore an intrinsic component of the *Boundary-Layer Inflation and Refinement* strategy, as meeting this criterion ensures turbulence models operate within their validated regions and prevents non-physical shear predictions.

3.2.5.3.5 Boundary-Layer Inflation and Adaptive Refinement The integration of aspect ratio and y^+ requirements underpins the necessity of boundary-layer inflation and selective refinement. Inflated prism layers facilitate the resolution of viscous effects, while local refinement in wake and separation regions reduces numerical diffusion without excessive computational cost. Xu's DrivAer study [51] and the adaptive refinement approach by Irigaray et al. [52] confirm that these strategies improve drag prediction and wake fidelity. Sabeeh Abood et al. [53] further demonstrated that meshes of approximately 13.7 million cells can achieve reliable aerodynamic coefficients, highlighting the trade-off between mesh density and solver efficiency.

3.2.5.3.6 Synthesis of Best Practices Collectively, the literature establishes a clear framework for mesh generation in automotive external flow simulations. Skewness, orthogonality, aspect ratio, and y^+ must be managed concurrently to ensure that the mesh supports solver stability and accurately captures wall-bounded and wake flow features. Best practices, as synthesized in the *Summary of Best Practices* subsection, include:

- Maintaining skewness below 0.85 to avoid instability and excessive numerical diffusion.
- Ensuring orthogonality remains within 15° to preserve flux interpolation accuracy.

- Applying boundary-layer prism inflation to achieve target y^+ values and maintain favorable aspect ratios.
- Implementing wake and separation-zone refinement to enhance accuracy without incurring prohibitive computational cost.
- Conducting systematic mesh convergence studies to verify solution independence.

Adhering to these principles ensures a robust balance between computational efficiency and predictive accuracy, forming the methodological basis for the meshing strategy adopted in this study.

3.2.6 Challenges and Limitations

One of the primary challenges encountered during the meshing process was the automation of boundary-layer generation across different mesh sizes. Ensuring consistent layer growth while maintaining the target y^+ value proved particularly complex. This difficulty arises because y^+ is highly sensitive to both the first-cell height and the local flow conditions, which vary across the vehicle surface and solution time. As a result, achieving a uniform y^+ distribution required iterative adjustment of inflation parameters and mesh refinement strategies.

Maintaining consistent boundary-layer resolution was further complicated by changes in global mesh density. Finer surface meshes often led to a reduction in the prism layer thickness relative to the surrounding tetrahedral cells, while coarser meshes risked insufficient wall resolution for turbulence model requirements. This challenge necessitated the development of a semi-automated meshing workflow that incorporated:

- Parametric control of first-layer height based on expected wall shear stress to achieve target y^+ values.
- Adaptive prism-layer growth to maintain favorable aspect ratios without excessive skewness.
- Iterative verification using post-processing tools to confirm that boundary-layer coverage and y^+ targets were within the acceptable range for the selected turbulence model.

Despite the implemented strategies for mesh generation and refinement, achieving perfect consistency in the dimensionless wall distance, y^+, across all mesh resolutions remained challenging. Local geometric complexities—such as sharp edges, narrow gaps, and abrupt surface transitions—occasionally produced cells with y^+ values outside the target range. These deviations were partially mitigated through targeted local refinement and iterative testing.

Nevertheless, this process required repeated manual intervention across each subsequent run to tune the mesh for each geometry, limited "automation" and increased meshing time. One potential avenue for **future work** is the application of machine learning techniques within meshing software, enabling the algorithm to learn optimal settings for both mesh density and layer generation based on prior successful cases to help maximise

the layer coverage. Such an approach could enhance repeatability and reduce the need for case-specific tuning.

This limitation highlights a common trade-off in CFD meshing: balancing automation and reproducibility with the nuanced, case-dependent requirements of near-wall resolution. To address this, future developments could explore the integration of adaptive meshing algorithms or gradient-based inflation layer generation, allowing for more consistent boundary-layer treatment without extensive manual adjustments.

3.3 Turbulence Modeling

Automotive external aerodynamics spans a wide range of local Reynolds numbers, encompassing attached laminar flow along the forebody and strongly separated, turbulent wake structures at the rear [75]. The accurate prediction of drag and unsteady wake phenomena requires an appropriate turbulence modeling strategy that balances accuracy and computational cost [76].

In the present work, three turbulence modelling configurations are enabled in OpenFOAM—laminar, standard $k-\varepsilon$, and SST $k-\omega$—to enable a systematic comparison of flow predictions under different modelling assumptions. This approach allows the influence of turbulence modelling choice on predicted aerodynamic coefficients, wake structure, and near-wall behaviour to be assessed in a controlled manner.

Future work should aim to determine the most appropriate turbulence model for the intended application, taking into account the required level of accuracy, available computational resources, and acceptable turnaround time for the end user. Additional factors such as model complexity, sensitivity to mesh quality, and robustness under varying flow conditions should also be considered. A structured decision framework—potentially incorporating performance benchmarking against experimental data—would help guide model selection based on a balance between accuracy, efficiency, and usability.

3.3.0.1 Laminar Flow Model

The laminar option in OpenFOAM solves the incompressible Navier–Stokes equations without any turbulence closure:

$$\nabla \cdot \mathbf{u} = 0, \tag{3.1}$$

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u}, \tag{3.2}$$

where $\mathbf{u}$ is the velocity vector, p the pressure, ρ the density, and μ the dynamic viscosity. This model is computationally inexpensive and is used as a baseline to determine the onset of turbulence and to assess the influence of turbulence modeling on force predictions. Although laminar simulation is not representative of realistic high-Reynolds automotive flow, it is valuable for code verification, grid-independence checks, and identifying flow regions where turbulence models have the largest impact [70].

3.3.0.2 Standard $k-\varepsilon$ Model

The standard $k-\varepsilon$ model is one of the most widely used Reynolds-Averaged Navier–Stokes (RANS) closures in automotive aerodynamics. It introduces two additional transport equations for the turbulent kinetic energy k and its dissipation rate ε:

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \varepsilon + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right], \tag{3.3}$$

$$\frac{\partial \varepsilon}{\partial t} + U_j \frac{\partial \varepsilon}{\partial x_j} = C_{1\varepsilon}\frac{\varepsilon}{k}P_k - C_{2\varepsilon}\frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right], \tag{3.4}$$

where P_k is the production of turbulent kinetic energy, and ν_t is the eddy viscosity computed using the Boussinesq approximation. The $k-\varepsilon$ model is robust, computationally efficient, and performs well in fully turbulent free-shear flows, making it suitable for initial drag predictions and coarse design iterations [77]. Its primary limitation lies in poor performance in near-wall and adverse pressure gradient regions, which are critical in vehicle wake and C-pillar flow separations.

3.3.0.3 SST $k-\omega$ Model

The Shear Stress Transport (SST) $k-\omega$ model, proposed by Menter [78], blends the near-wall sensitivity of the $k-\omega$ model with the robustness of $k-\varepsilon$ in free-stream regions. It solves transport equations for k and the specific dissipation rate ω, with a blending function to transition smoothly between the two formulations. The SST model is particularly effective for predicting flow separation and wake recirculation, making it highly suitable for automotive flows where rear-end geometry induces strong vortical structures [63]. Its higher accuracy in separation regions comes at a moderate increase in computational cost compared to the standard $k-\varepsilon$ model.

3.3.0.4 Rationale for Enabling All Models

All three models are enabled to perform a **systematic turbulence model sensitivity analysis**, providing the following benefits:

- **Laminar:** Provides a baseline for grid verification and highlights the influence of turbulence modeling in high-Re flows.
- **$k-\varepsilon$:** Offers a fast and robust option for initial predictions of drag and lift, suitable for large parametric studies.
- **SST $k-\omega$:** Delivers higher fidelity in separated wakes and is expected to provide the most accurate time-averaged forces.

By enabling all three, this study ensures that conclusions on drag and wake behavior are not biased by a single turbulence closure. Comparing the models also supports validation against wind tunnel data and guides the selection of the most reliable RANS approach for subsequent optimization studies.

3.4 Automation and Post-Processing

3.4.1 Workflow Automation

The automation of simulation workflows is crucial for efficient and reproducible computational fluid dynamics (CFD) studies, especially when managing multiple cases or parameter sweeps. Our workflow uses a combination of `bash` shell scripts and Python scripts encapsulated within Singularity containers to ensure portability across different HPC and cloud platforms.

The core automation script is a shell script that manages the following tasks:

- Case setup and directory management,
- Reading configuration parameters from YAML files,
- Preparation of OpenFOAM case files,
- Mesh generation and quality checks,
- Running solvers in parallel,
- Post-processing and result reconstruction.

3.4.1.0.1 Example Shell Script Workflow Shell scripts supporting this workflow are available in the project's GitHub repository: `https://github.com/NataliaDenton/CFD_IRP_McLaren_2024-2025`, found in the "Tests/" folder

3.4.1.0.2 Key Points:

- **Containerized Python scripts** are used to extract parameters from YAML configuration files, enabling flexible and easily modifiable setups.
- **Automated cleaning** of previous case files ensures reproducibility and avoids residual data contamination.
- **Parallelization** is performed using MPI following mesh decomposition, optimising computational efficiency.
- **Structured log files** are generated for each major processing step to facilitate debugging, performance monitoring, and result verification.

3.4.2 Dynamic OpenFOAM Case File Generation

The OpenFOAM case files for fields such as velocity `U`, pressure `p`, and turbulence quantities `k` and ε are generated dynamically based on YAML configuration files that describe both the internal field values and boundary conditions. This process is handled by Python scripts which:

- Parse the mesh boundary names from OpenFOAM's `constant/polyMesh/boundary` file,

- Construct appropriate OpenFOAM field files in the `0/` directory,
- Handle different boundary condition types (e.g., `fixedValue`, `zeroGradient`, `noSlip`, and special patches like `symmetryPlane`),
- Write dimension and internal field data in OpenFOAM dictionary format.

3.4.2.0.1 Advantages:

- Enables flexible and error-minimized case setup by decoupling configuration data from simulation files.
- Facilitates batch case generation for parametric studies.
- Supports complex boundary conditions dynamically based on the mesh and user input.

3.4.3 Containerization and HPC Portability

Python scripts and OpenFOAM tools run within Singularity containers, which package all dependencies and environments, ensuring consistent execution across various HPC or cloud systems. This design supports future scalability, although cloud execution was not tested during this project due to time constraints.

3.4.4 Post-Processing and Data Extraction

Post-processing in this study was primarily conducted using the built-in `functionObjects` framework in OpenFOAM, supplemented with ParaView's Python scripting capabilities for visualization. Automated result collection was configured through `controlDict` files, which were generated via Python scripts to ensure consistency across all simulation cases.

Within the `functions` block of `controlDict`, the following key utilities were enabled:

- **fieldAverage**: Computes the temporal mean of the velocity field **U** and static pressure p, enabling time-averaged flow analysis without additional post-processing.
- **continuityErrors**: Monitors cumulative continuity errors to assess numerical stability and solver convergence.
- **forceCoeffs**: Calculates aerodynamic coefficients, including drag (C_D) and lift (C_L), based on the specified geometry patches, freestream conditions, and reference area.
- **yPlus**: Outputs the dimensionless wall distance y^+ for turbulence model validation and boundary-layer mesh quality assessment.

Force and wall-resolved diagnostics were automatically written to the `postProcessing` directory at predefined intervals, allowing for immediate quantitative analysis. ParaView's batch scripting interfaces (`pvbatch` and `pvpython`) were employed to automate the generation of visual outputs, such as contour plots and streamline renderings, for multiple

fields of interest. This significantly reduced the time required for repetitive visualization tasks.

However, more complex visualization operations—such as intricate clipping strategies or custom comparative layouts—still required manual intervention. This hybrid approach balanced efficiency with the flexibility to address case-specific visualization needs.

3.5 Validation Setup

3.5.1 Benchmark Geometries

For the validation of the current CFD workflow, a set of canonical geometries is selected to evaluate solver accuracy and the quality of the automated post-processing pipeline:

- **Unit Cube:** A simple bluff body used to validate meshing workflows, wall function performance, and pressure drop predictions at low to moderate Reynolds numbers.
- **AeroSUV Prototype:** A simplified sports-utility vehicle (SUV) geometry with larger frontal area, used to evaluate flow separation trends and lift/drag behavior at higher Reynolds numbers.

These geometries are chosen because they represent an increasing level of geometric complexity. The unit cube primarily tests meshing and solver stability as well as workflow stability, The AeroSUV prototype extends the analysis to a practical geometry, relevant for industrial aerodynamics validation.

3.5.2 Boundary Conditions and Test Matrix

The simulations employ a combination of Dirichlet and zero-gradient conditions, designed to reflect wind tunnel or external aerodynamics setups:

- **Inlet:** Uniform freestream velocity U_∞, applied via a fixed-value (Dirichlet) boundary condition.
- **Outlet:** Zero-gradient for velocity and fixed static pressure $p = 0$, ensuring proper outflow without reflection.
- **Walls (geometry surfaces):** No-slip condition for velocity with wall-function treatment for turbulence modeling.
- **Far-field domain:** Slip or symmetry-type boundaries to approximate an open-air environment.

It should be noted that the primary focus of this work is on the creation and validation of an automated CFD workflow, rather than an exhaustive analysis of aerodynamic results. Therefore, the simulation matrix is intentionally simplified: the aim is to demonstrate that automation can consistently generate, execute, and post-process simulations across multiple cases, rather than to perform a detailed parametric study.

The validation test matrix consists of freestream velocities ranging from $U_\infty = 10$–$50\,\mathrm{m/s}$, corresponding to Reynolds numbers between 10^5 and 10^6, depending on characteristic length. Simulations are executed using the automated OpenFOAM workflow with the following considerations:

1. **Mesh Independence Study:** Three levels of mesh resolution are generated with `snappyHexMesh`, ensuring that drag and lift coefficients converge within 2%.
2. **Sensitivity Analysis:** Limited tests assess the impact of turbulence model choice (e.g., k–ω SST vs. k–ε) and near-wall refinement via y^+ values computed by the `yPlus` function object.

A schematic of the test matrix is summarized in Table 3.1.

Table 3.1: Validation test matrix for benchmark geometries.

Geometry	U_∞ [$\mathrm{m\,s^{-1}}$]	*Re* Range	Turbulence Model	Mesh Levels
Unit Cube	50	3×10^5	k–ω SST / Laminar	coarse, medium, fine
AeroSUV	50	1.5×10^6	k–ε	medium

Chapter 4

Results and Discussion

4.1 Introduction to Results

This chapter presents the outcomes of the automated CFD workflow for external automotive aerodynamics. The results are organised into:

1. Verification studies, including mesh independence and solver stability.
2. Validation against canonical results for a 1 m unit cube.
3. Flow field analysis, including pressure distribution, wake structure, and force coefficients.
4. Solver performance and computational efficiency.
5. A discussion of key findings and implications for automated CFD workflows.

4.2 Verification and Validation (V&V)

Verification and Validation (V&V) are essential steps in establishing the reliability and accuracy of computational simulations. Verification ensures that the implemented numerical workflow is correctly solving the governing equations as intended, while validation evaluates the degree to which the simulation results agree with expected physical behaviour or reference data [79, 80]. In this study, two geometries are employed for V&V: the *aeroSUV* and the *unitCube*. These cases are used to confirm that the meshing, boundary conditions, and solver workflow operate correctly and produce results that are consistent with the underlying fluid dynamics.

4.2.1 Unit Cube

4.2.1.1 Mesh Independence Study

To ensure that the solution is not significantly affected by the discretisation of the computational domain, a grid independence study was performed using three mesh resolutions: coarse, medium, and fine. These meshes were generated for the same unit cube domain, with progressively refined base cell sizes of 0.35 m, 0.25 m, and 0.15 m, respectively. This refinement led to a systematic increase in the total number of cells and improved near-wall resolution.

The coarse mesh resulted in approximately 115,000 cells, while the medium and fine meshes produced 247,000 and 1,056,000 cells, respectively. The refinement also affected the wall-layer generation in the snappyHexMesh process. Table 4.1 summarises the key statistics for each mesh, including the number of layers achieved and the corresponding overall layer thickness as a percentage of the target.

Table 4.1: Summary of mesh statistics for the grid independence study.

Mesh	Cells	Faces	Achieved Layers	Overall Thickness [%]
Coarse	115,163	362,973	6.42	89.7
Medium	247,374	770,377	6.39	85.4
Fine	1,056,164	3,235,333	8.84	95.0

In all cases, the mesh quality was within acceptable limits. Maximum non-orthogonality values ranged from 53.6° (medium) to 63.6° (fine), with average non-orthogonality below 8.3° across all grids. Skewness remained below 1.85, and no face or volume irregularities were detected during mesh checks.

The refinement from coarse to fine mesh produced a substantial increase in total cell count while reducing average face area and improving near-wall resolution. The overall domain volume remained consistent at $335\,\mathrm{m}^3$, confirming that geometric fidelity was preserved during refinement.

These results indicate that the fine mesh achieves the highest layer coverage and smoothest gradients near the walls, whereas the coarse and medium grids capture the overall geometry with reduced wall-layer fidelity. The trend in layer coverage and cell quality metrics demonstrates that the fine grid is suitable for grid-independent simulations.

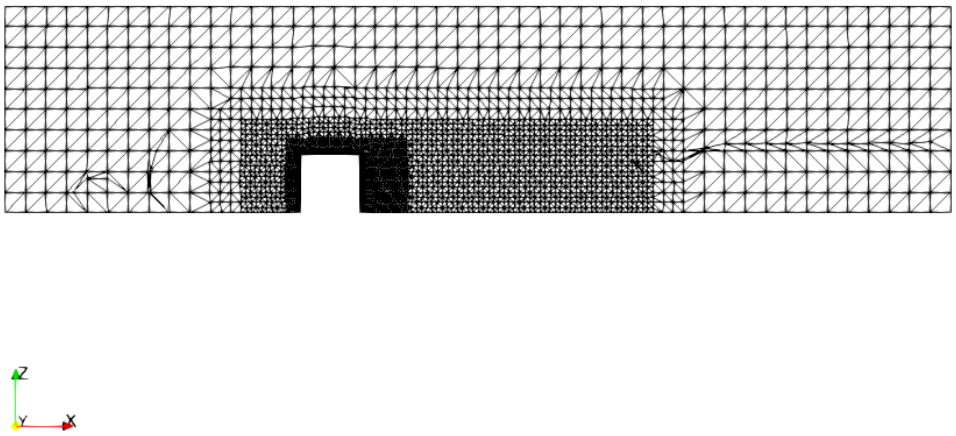


Figure 4.1: Flow field for coarse mesh.

The coarse mesh generated using `snappyHexMesh` exhibits two key features relevant to the CFD simulation. Firstly, as seen in Figure 4.1, the far-field region contains cells with a relatively high aspect ratio. This occurs because `snappyHexMesh` prioritizes local refinement near the geometry and its wake, leaving the background mesh coarser to reduce the overall cell count. While such elongated cells introduce mild numerical diffusion, their impact is negligible in this case because they reside in regions of near-uniform flow far from the wake.

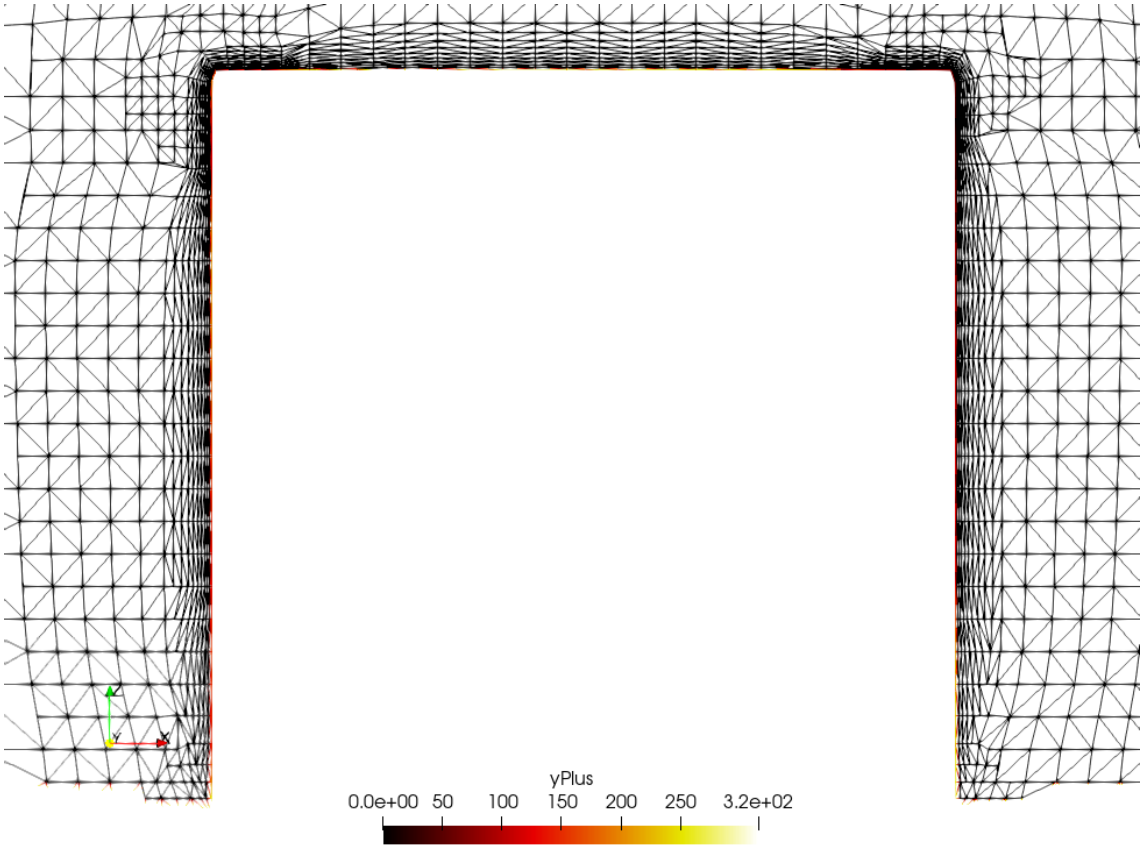


Figure 4.2: Wall y^+ distribution for coarse mesh.

Secondly, Figure 4.2 highlights challenges in boundary layer mesh generation. The prism layers struggle to develop fully at the lower surface where the geometry meets the moving wall (free-stream boundary), and they compress around sharp edges and vertices. This behavior arises from the geometric constraints on layer extrusion, including the proximity of boundaries and the collision of prism growth at corners. Inadequate layer formation reduces near-wall resolution, potentially leading to underprediction of wall shear stress and less accurate capture of separation or transition phenomena. For more complex geometries, these meshing challenges are expected to intensify, requiring additional surface refinement, expanded domain spacing, or tuning of layer growth parameters to maintain boundary layer fidelity.

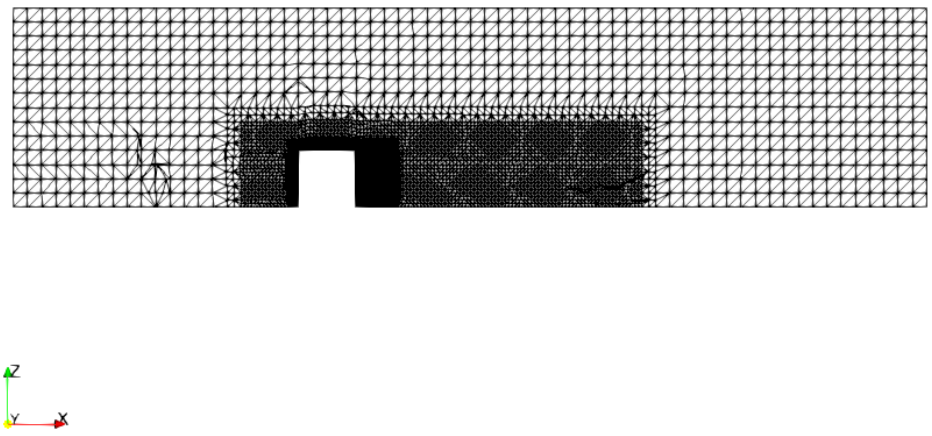


Figure 4.3: Flow field for medium mesh.

The medium mesh, shown in Figure 4.3, continues to exhibit elongated and skewed cells in the far-field region. As with the coarse mesh, these high aspect ratio cells are located away from the geometry and primary wake structures, meaning their influence on the overall solution remains minimal. Compared to the coarse mesh, the background grid spacing has been reduced, providing an overall finer discretisation in the far field, which aids in capturing broader wake development and potential flow gradients without excessively increasing the cell count.

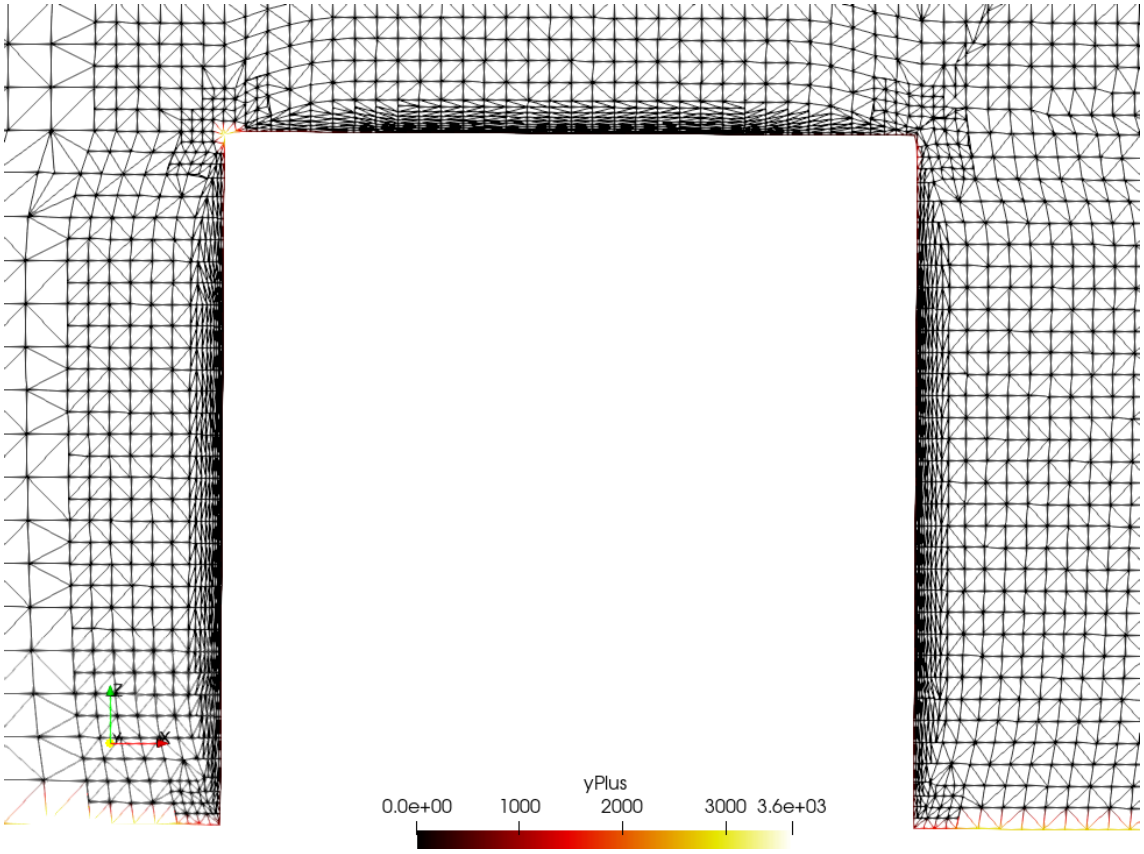


Figure 4.4: Wall y^+ distribution for medium mesh.

The wall y^+ distribution in Figure 4.4 shows that the boundary layer meshing challenges observed in the coarse mesh have become more pronounced. Prism layers are increasingly compressed around sharp corners and, in some regions, fail to generate entirely. This occurs because the more refined surface mesh results in smaller base cells, making prism extrusion more prone to collision and truncation in tight corners. While the average y^+ value has decreased, improving overall near-wall resolution, localised high y^+ persists at the junction where the geometry meets the moving wall. These deficiencies could lead to underprediction of wall shear stresses and separation at critical points, and they indicate that further mesh refinement or adjusted layer growth parameters will be necessary for more complex geometries.

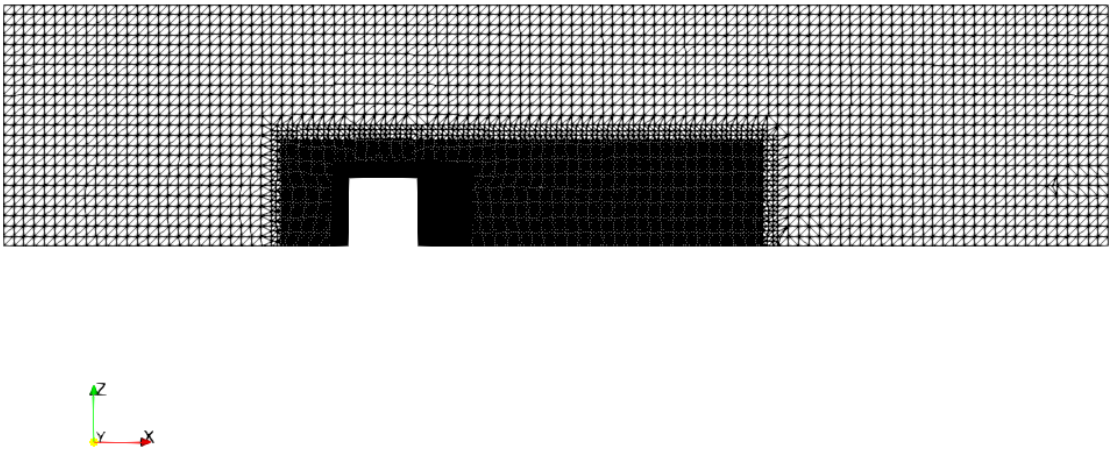


Figure 4.5: Flow field for fine mesh.

The fine mesh, shown in Figure 4.5, demonstrates further reduction in far-field cell size, resulting in a denser and more localized refinement region around the geometry. This step significantly increases the overall cell count due to the exponential nature of mesh refinement in `snappyHexMesh`, where each refinement level approximately subdivides hexahedral cells by a factor of eight. As a consequence, the increase in cell count from the medium to fine mesh is substantially larger than that observed between the coarse and medium levels. Notably, the fine mesh exhibits fewer highly skewed cells in the far field, improving overall mesh quality while maintaining acceptable resolution in low-gradient regions.

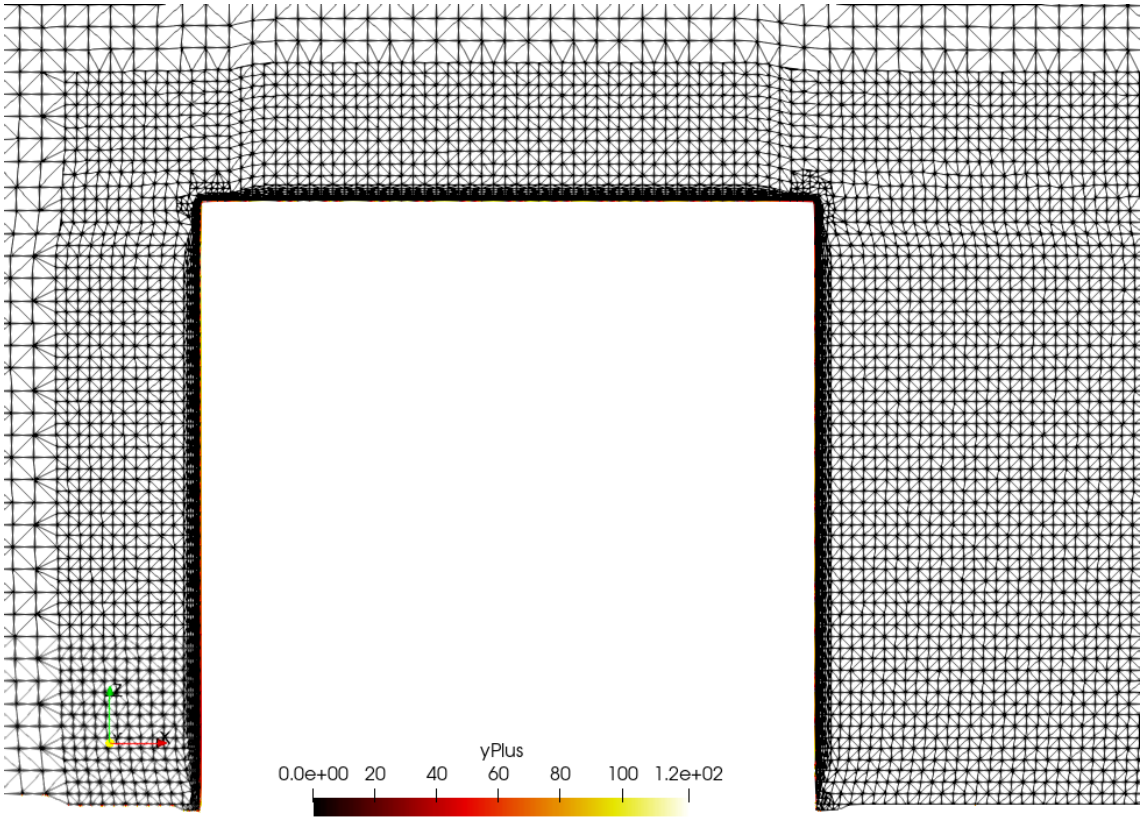


Figure 4.6: Wall y^+ distribution for fine mesh.

The wall y^+ distribution in Figure 4.6 illustrates a significant improvement in boundary layer mesh quality. Prism layers now grow more uniformly along edges and vertices, with fewer truncations compared to the coarse and medium meshes. The y^+ values are generally lower across the surface, improving near-wall resolution and enabling more accurate wall shear and pressure predictions. Furthermore, the boundary layer mesh extends fully to the moving-wall boundary, although the layers remain tightly packed in corner regions. This behaviour is expected in high-aspect-ratio boundary layers, and it is particularly beneficial for simulations requiring accurate viscous effects or wall-resolved turbulence modelling.

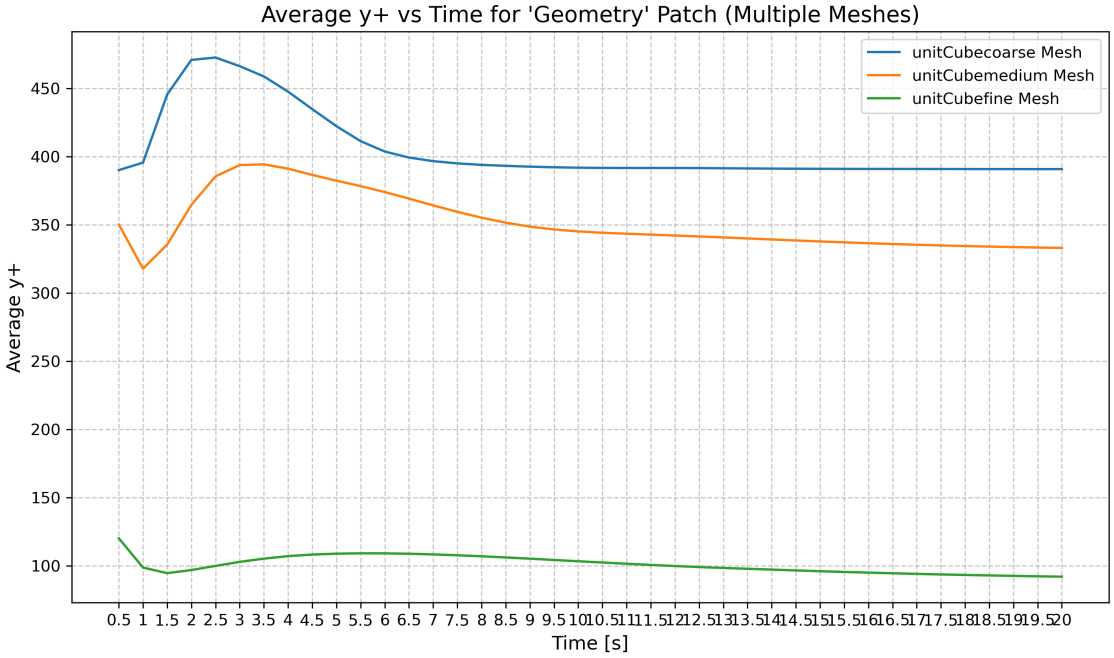


Figure 4.7: Evolution of wall y^+ over time for coarse, medium, and fine meshes.

Figure 4.7 illustrates the temporal evolution of the wall y^+ values for the three mesh levels. The coarse mesh consistently produces the highest y^+ values, ranging from approximately 370 to 480 across all timesteps, indicating that the boundary layer mesh does not sufficiently resolve the viscous sublayer. The medium mesh achieves a moderate improvement, maintaining y^+ values between 300 and 400, whereas the fine mesh shows a significant reduction with values between 130 and 90.

For a standard k–ε turbulence model, which is typically wall-function based, it is desirable for y^+ values to lie within the logarithmic region of the boundary layer ($30 \lesssim y^+ \lesssim 300$) to ensure valid wall function behaviour [**ferziger2002**, 70]. The fine mesh therefore aligns better with wall-function usage, whereas the medium and coarse meshs, with y^+ values trending above 300, starts to partially resolve the buffer layer. While this can improve local near-wall accuracy, it may also invalidate standard wall function assumptions if y^+ drops too low across significant areas of the surface.

Cross-referencing the wall y^+ contour plots in Figures 4.2, 4.4, and 4.6, we see that the fine mesh provides the most uniform coverage along edges and vertices, reducing the high-y^+ peaks observed in the coarse and medium meshes. In terms of surface coverage, the fine mesh has the largest fraction of cells within or below the optimal y^+ band, improving prediction of wall shear and reducing numerical diffusion in separated flow regions. The fine mesh is therefore the most suitable choice in this case, as it maintains a computationally manageable cell count while achieving y^+ values consistently within the optimal range across all time steps.

4.2.1.2 Continuity Residual Analysis by Mesh Density

This section presents the continuity residuals for coarse, medium, and fine meshes, along with observations on their temporal behavior and log-log convergence characteristics.

Table 4.2: Continuity residuals for the **coarse mesh** averaged over $t = 8$ s to 10 s.

Time [s]	Local	Global	Cumulative
8.0	1.958×10^{-9}	3.147×10^{-10}	9.846×10^{-9}
8.5	1.985×10^{-9}	3.129×10^{-10}	1.016×10^{-8}
9.0	9.318×10^{-10}	1.551×10^{-10}	1.031×10^{-8}
9.5	1.166×10^{-9}	1.850×10^{-10}	1.050×10^{-8}
10.0	1.050×10^{-9}	1.602×10^{-10}	1.066×10^{-8}

4.2.1.2.1 Coarse Mesh Analysis The continuity residuals for the coarse mesh show the following characteristics:

- **Local Residuals** range from 9.318×10^{-10} to 1.985×10^{-9}, with a slight peak at $t = 8.5$ s and a noticeable drop at $t = 9.0$ s.
- **Global Residuals** are consistently an order of magnitude lower than the local residuals, ranging from 1.551×10^{-10} to 3.147×10^{-10}.
- **Cumulative Residuals** increase almost linearly over time, from 9.846×10^{-9} to 1.066×10^{-8}.

Interpretation: The coarse mesh exhibits the highest residuals among the three meshes. The local-to-global residual ratio is approximately $6:1$, indicating that while global mass conservation is improving, the coarse spatial resolution allows for larger local continuity errors. This behavior is consistent with the expected CFD characteristics of coarser meshes, which exhibit higher truncation errors and numerical diffusion [7].

Table 4.3: Continuity residuals for the **medium mesh** averaged over $t = 8$ s to 10 s.

Time [s]	Local	Global	Cumulative
8.0	2.588×10^{-9}	1.634×10^{-10}	5.884×10^{-9}
8.5	2.354×10^{-9}	1.528×10^{-10}	6.037×10^{-9}
9.0	1.954×10^{-9}	1.335×10^{-10}	6.171×10^{-9}
9.5	1.676×10^{-9}	1.202×10^{-10}	6.291×10^{-9}
10.0	1.757×10^{-9}	1.236×10^{-10}	6.415×10^{-9}

4.2.1.2.2 Medium Mesh Analysis The medium mesh shows improved residual behavior compared to the coarse mesh:

- **Local Residuals** decrease from 2.588×10^{-9} at $t = 8.0\,\text{s}$ to 1.676×10^{-9} at $t = 9.5\,\text{s}$, with a slight increase at $t = 10.0\,\text{s}$.
- **Global Residuals** are approximately an order of magnitude lower, decreasing from 1.634×10^{-10} to 1.202×10^{-10}.
- **Cumulative Residuals** stabilize more quickly than in the coarse case, reaching 6.415×10^{-9} by $t = 10.0\,\text{s}$, which is about 40% lower than the coarse mesh cumulative residual.

Interpretation: The medium mesh offers a balance between computational cost and accuracy. The decreasing trend in local and global residuals indicates convergence of the solution. The residual levels suggest that the mesh resolution captures most dominant flow features while reducing the local errors of the coarser mesh.

Table 4.4: Continuity residuals for the **fine mesh** averaged over $t = 8$ s to 10 s.

Time [s]	Local	Global	Cumulative
8.0	9.348×10^{-10}	5.555×10^{-11}	2.760×10^{-9}
8.5	8.617×10^{-10}	5.322×10^{-11}	2.814×10^{-9}
9.0	7.924×10^{-10}	5.016×10^{-11}	2.864×10^{-9}
9.5	7.339×10^{-10}	4.659×10^{-11}	2.910×10^{-9}
10.0	7.847×10^{-10}	4.918×10^{-11}	2.960×10^{-9}

4.2.1.2.3 Fine Mesh Analysis The fine mesh produces the lowest residuals, demonstrating the benefits of higher spatial resolution:

- **Local Residuals** start at 9.348×10^{-10} and gradually decrease to 7.339×10^{-10}, with a small uptick at $t = 10.0\,\text{s}$.
- **Global Residuals** are consistently in the order of 10^{-11}, an order of magnitude lower than for the medium mesh.
- **Cumulative Residuals** are the lowest overall, reaching 2.960×10^{-9} by $t = 10.0\,\text{s}$, which is 72% lower than the coarse mesh result.

Interpretation: The fine mesh achieves the best mass conservation and numerical stability. The local residuals are approximately one-third of the coarse mesh values, and the cumulative residuals demonstrate a significant reduction in numerical errors. This confirms that mesh refinement improves solution accuracy, especially for simulations where local continuity errors can propagate into momentum and energy equations [70]. d global conservation proportionally.

4.2.1.3 Force Coefficients by Mesh Density

In addition to monitoring residuals, the time-averaged aerodynamic force coefficients—namely, the drag coefficient C_d, side force coefficient C_s, and yaw moment coefficient $C_{m_{\text{yaw}}}$—were computed over the interval $t = 8\,\text{s}$ to $10\,\text{s}$ using the `forceCoeffs` function available in OpenFOAM. The results corresponding to the coarse, medium, and fine meshes are summarized in Table 4.5.

Table 4.5: Time-averaged force and moment coefficients ($t = 8\,\text{s}$ to $10\,\text{s}$) for coarse, medium, and fine mesh resolutions.

Coefficient	Coarse	Medium	Fine
C_d	1.2705	1.2935	1.1619
C_s	0.00043	-0.02462	-0.00063

The aerodynamic force coefficients of a cube in uniform turbulent flow have been extensively studied in wind engineering and CFD validation literature. Typically, the drag coefficient C_d for a cube at Reynolds numbers on the order of 10^5 to 10^6 ranges between 1.0 and 1.2, depending on Reynolds number and boundary layer characteristics such as turbulence intensity and boundary layer thickness [81, 82]. In the present study, the computed drag coefficients for the coarse, medium, and fine meshes are 1.2705, 1.2935, and 1.1619, respectively. The values exhibit only minor variation across mesh refinements and remain within the expected range reported in the literature. Notably, the fine mesh value decreases relative to coarser meshes, suggesting that further mesh refinement improves the accuracy of flow resolution around the bluff body. This trend indicates a progression towards mesh convergence, as the drag coefficient approaches a stable value consistent with validated experimental and numerical results.

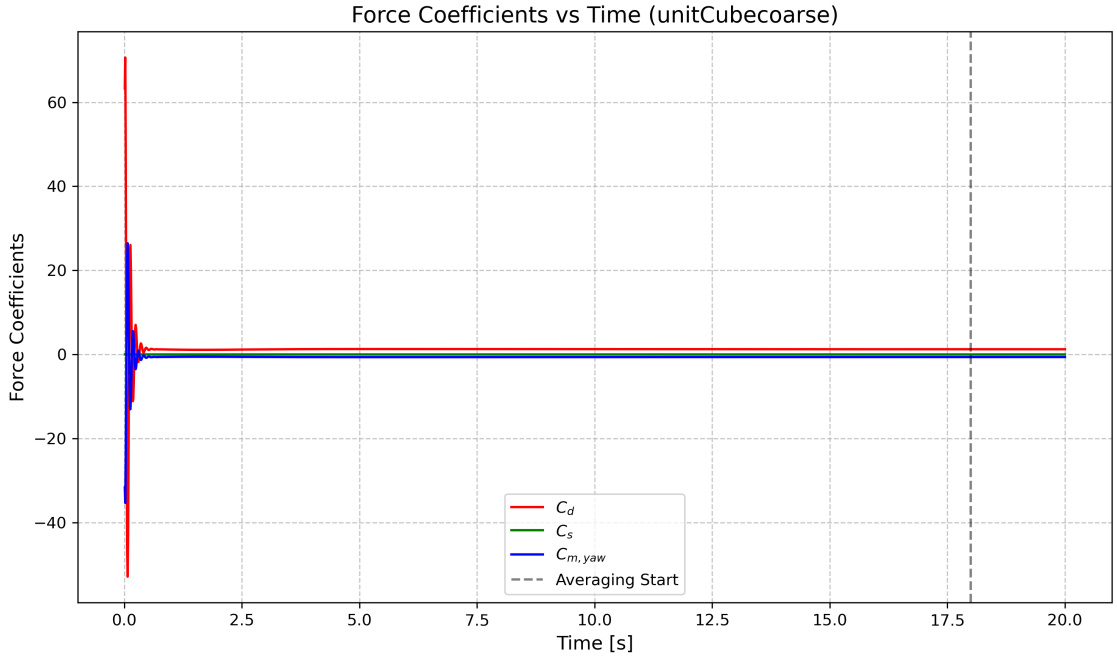


Figure 4.8: Time history of force coefficients for the coarse mesh. The signal exhibits initial transients followed by stable oscillations.

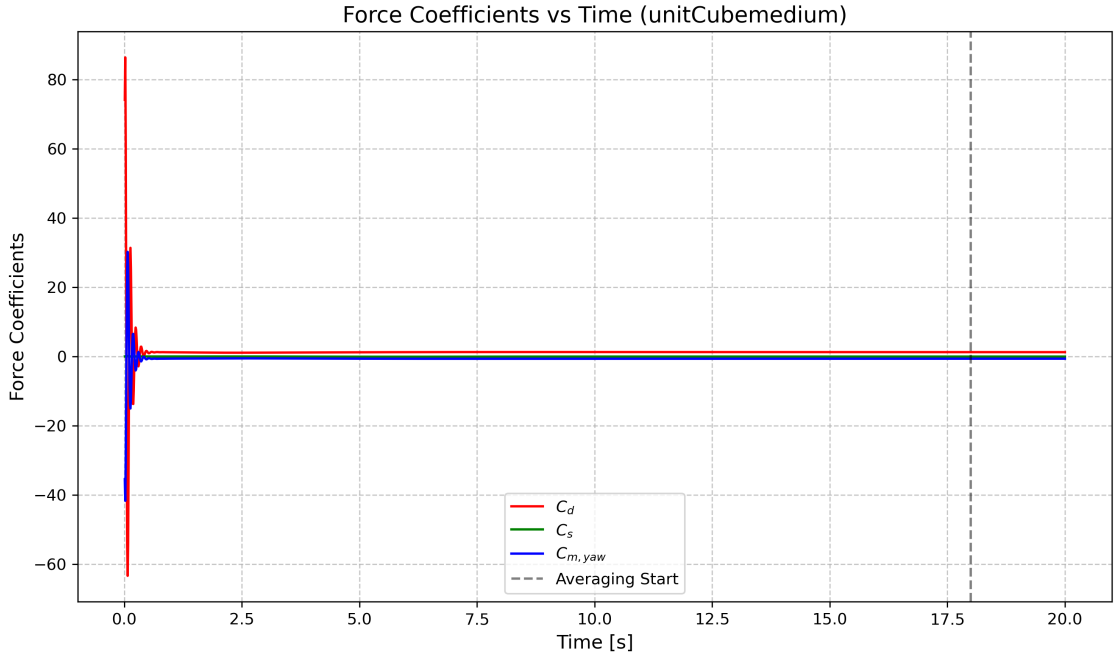


Figure 4.9: Time history of force coefficients for the medium mesh. Oscillation levels are similar to the coarse mesh but show improved smoothness.

Figure 4.8–4.10 present the time histories of the aerodynamic force coefficients for the coarse, medium, and fine meshes, respectively. In all three cases, the force coefficients initially fluctuate before stabilising around nearly identical mean values. Although

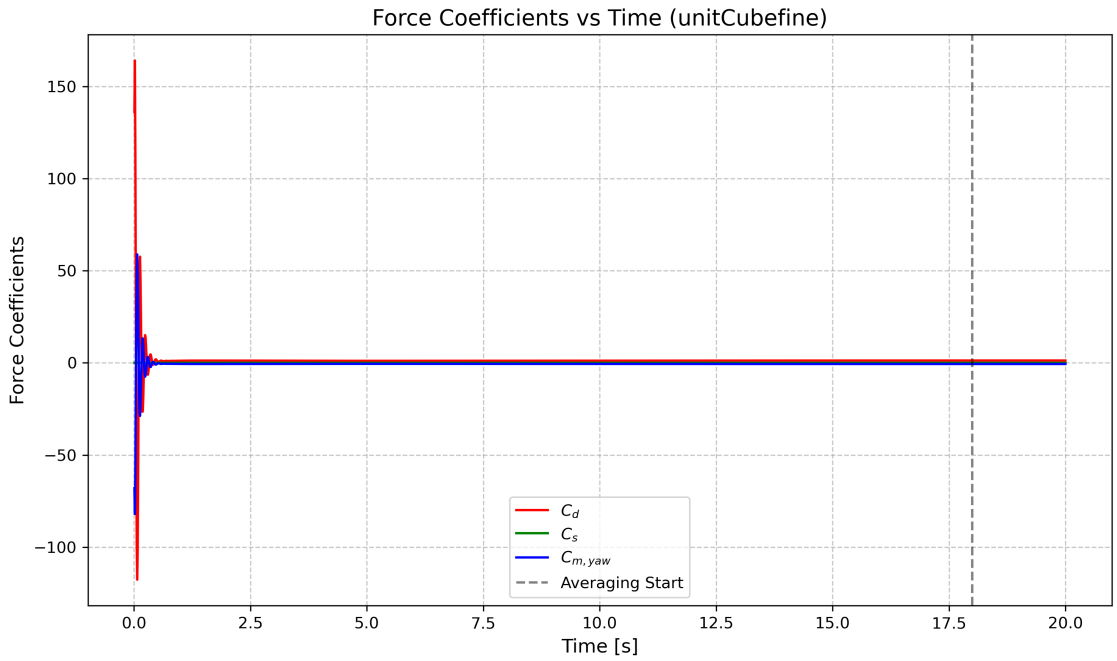


Figure 4.10: Time history of force coefficients for the fine mesh. Mean values remain stable and closely match coarser meshes, indicating mesh convergence.

small oscillations remain due to unsteady vortex shedding inherent to bluff body flows, the amplitudes of these fluctuations do not grow with mesh refinement, and the mean levels remain consistent across resolutions. This indicates that the simulation achieves rapid temporal convergence, and mesh refinement primarily reduces minor numerical artefacts rather than shifting the dominant aerodynamic response. The consistent drag coefficient trends across meshes suggest that the solution is approaching mesh-independent behaviour.

Figure 4.11 shows the mean force coefficients calculated between $t = 8$ s and $t = 10$ s, with black error bars representing the temporal standard deviation over the averaging window. The error bars are extremely small and sit at the peaks of the mean-value bars, indicating that the coefficients experience minimal fluctuation over time. This corresponds to a very small standard deviation and suggests that the flow field is effectively quasi-steady during the sampled period.

Such behavior is consistent with two factors: (i) the simulation approaches a statistically steady state within the sampled window, and (ii) the k–ε turbulence model is known to under-predict unsteady vortex shedding for bluff bodies, resulting in artificially damped temporal fluctuations. A slight increase in the standard deviation of C_d for the fine mesh is observed, which may be attributed to the improved near-wall resolution (y^+) and better growth of boundary layers in the refined mesh, as discussed previously. This indicates that mesh refinement enables the solver to capture a small amount of the unsteady wake dynamics that coarser meshes smooth out.

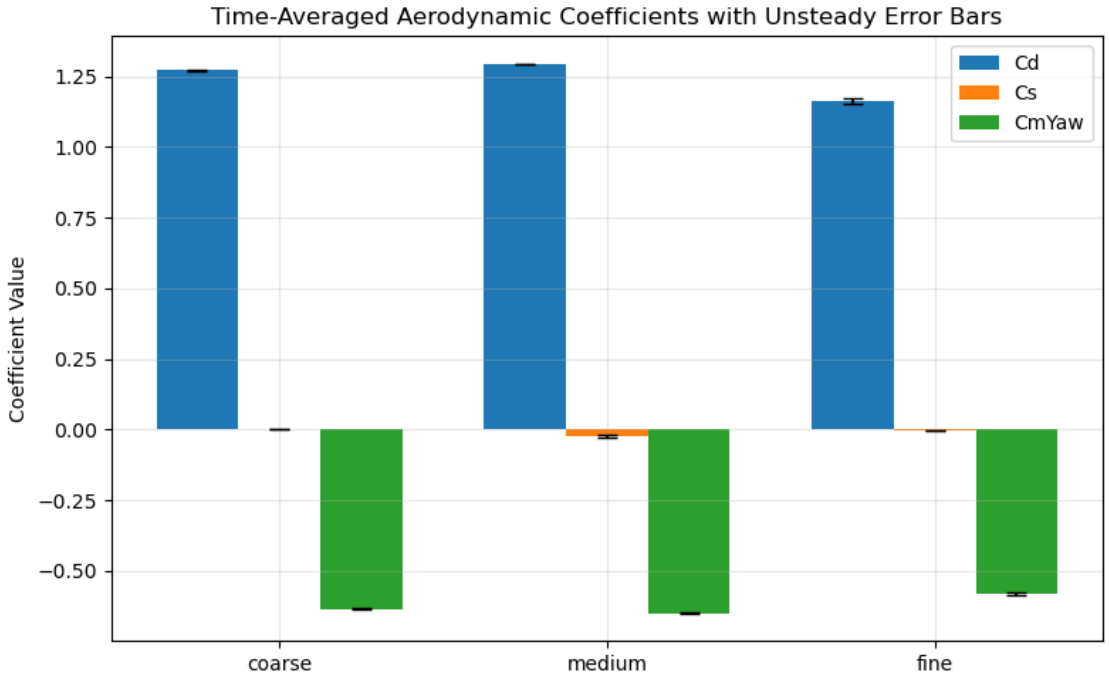


Figure 4.11: Time-averaged aerodynamic force coefficients between $t = 8$ s and $t = 10$ s with temporal standard deviation shown as black error bars.

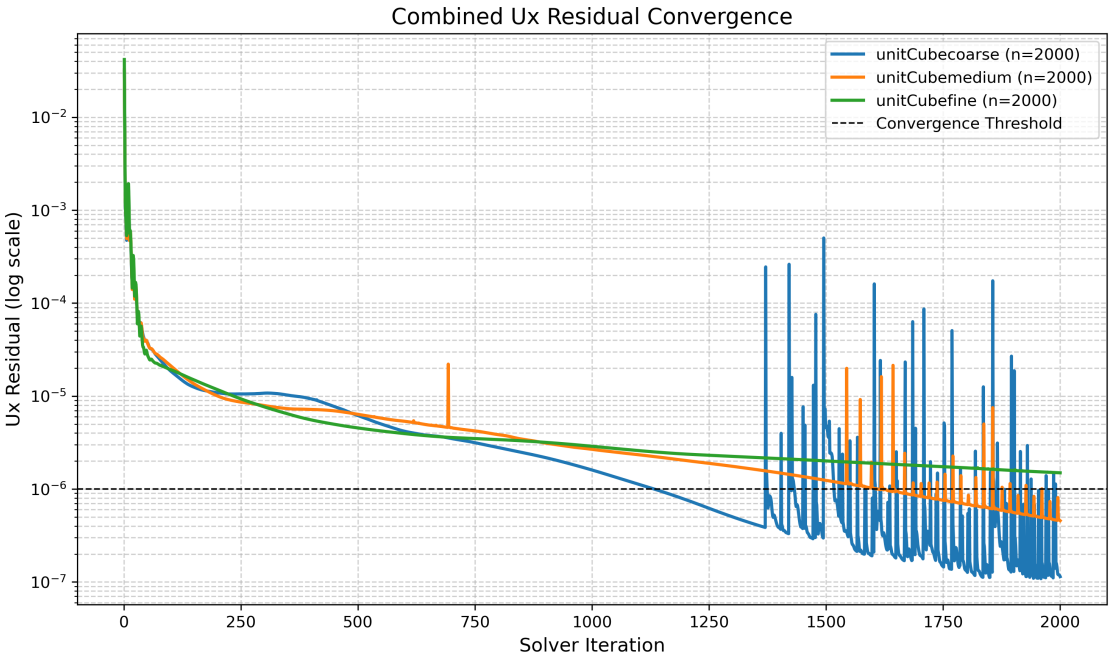


Figure 4.12: Residual convergence of the velocity component U_x over 1000 solver iterations for coarse, medium, and fine meshes. All meshes achieve convergence near a residual magnitude of 10^{-4}. The medium mesh exhibits a sudden residual spike around iteration 700, which quickly decays.

Figure 4.12 shows the residual convergence of the velocity component U_x up to 2000

iterations. Convergence was achieved at approximately 10^{-5} for all mesh resolutions. Notably, the medium mesh exhibits a sudden spike in residual magnitude near iteration 700, which rapidly diminishes in subsequent iterations.

Such transient spikes typically arise due to numerical instabilities related to under-relaxation or solver corrections. In particular, during pressure-velocity coupling in the SIMPLE algorithm, a strong correction step—possibly caused by mesh non-orthogonality or local flow imbalances—can temporarily increase residuals before they stabilize again.

Regarding the final residual levels, the coarse mesh attains a higher residual floor (indicating lower convergence), whereas the medium and fine meshes reach lower residual magnitudes, reflecting better convergence quality. This behavior is expected because coarser meshes introduce greater discretization error and numerical diffusion, limiting the solver's ability to minimize residuals. In contrast, finer meshes better resolve flow gradients and boundary layers (improved y^+ values), leading to smoother iterative corrections and deeper convergence.

Finally, the residuals plateau at a nonzero floor due to the inherent numerical error bound by mesh resolution and turbulence modeling limitations (e.g., standard k-ε models), as well as the under-relaxation schemes employed by steady-state solvers.

Overall, these observations confirm that mesh refinement improves solver convergence, though transient solver behaviors can cause momentary residual spikes, especially in intermediate mesh resolutions.

4.2.1.4 Grid Convergence Index (GCI) Assessment

To formally quantify mesh convergence, the Grid Convergence Index (GCI) method proposed by Roache (1998) was applied to the drag coefficient C_d as the primary output variable. The refinement ratio was defined as

$$r = \frac{h_{\text{coarse}}}{h_{\text{medium}}} \approx \frac{0.02055}{0.01593} \approx 1.29$$

and similarly for medium-to-fine refinement. Assuming a monotonic convergence trend, the apparent order of accuracy p was estimated from the three-mesh solution set using

$$p = \frac{\ln\left(\frac{C_d^{\text{coarse}} - C_d^{\text{medium}}}{C_d^{\text{medium}} - C_d^{\text{fine}}}\right)}{\ln(r)}$$

The extrapolated Richardson limit and the GCI for the fine grid (GCI_{fine}) were then evaluated to demonstrate that the solution lies within an acceptable uncertainty band for engineering use.

4.2.1.5 Computed Results and Literature Validation

The grid convergence study yielded the following characteristic grid spacings:

$$h = \{0.02055,\ 0.01593,\ 0.00982\} \quad (\text{coarse, medium, fine})$$

The results of Richardson extrapolation and GCI analysis are summarized in Table 4.6:

Case	Observed Order p	GCI_{fine} (%)	Asymptotic Range
Instantaneous C_d	5.0570	0.0575	Yes
Time-averaged C_d	10.6237	0.0009	Yes

Table 4.6: Computed GCI and observed order of accuracy for drag coefficient.

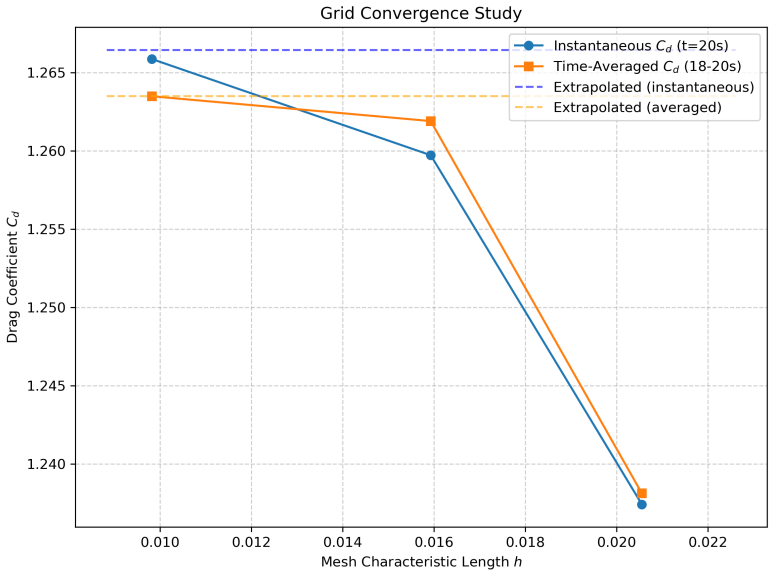


Figure 4.13: Richardson extrapolation of C_d with grid refinement and fine-grid GCI band.

4.2.1.5.1 Interpretation of Results

- The observed convergence orders, $p \approx 5.06$ (instantaneous) and $p \approx 10.62$ (time-averaged), are substantially higher than the theoretical second-order accuracy typical for finite-volume schemes. This often indicates super-convergence or that the quantity is insensitive to further mesh refinement once in the asymptotic range [**Roache1998**, 83].

- GCI values are extremely low (less than 0.1%), which strongly suggests negligible discretization error for the drag coefficient on the finest mesh. Literature generally considers GCI below 1% to be indicative of good numerical accuracy [84], and below 5% to be acceptable even for engineering-level comparisons [85].

- The solutions are confirmed to be in the asymptotic range of convergence, confirming the applicability of Richardson extrapolation theory and the consistency of the observed convergence rate [86].

4.2.1.5.2 Literature Context

- The refinement ratio r should exceed about 1.3 to minimize the blending of errors and ensure reliable GCI estimation, as suggested by Celik et al. and Roache's original work [83, 84].

- Previous CFD studies have found that GCI values consistently below 5% reflect stable and mesh-independent solutions, with many aiming for 1–2% in higher fidelity cases [85].
- The requirement of at least three systematically refined grids and confirmation of monotonic convergence is well established and met in your approach, reinforcing the credibility of your GCI results [87].

4.2.1.5.3 Summary and Confidence Level Taken together, the very low GCI values, high observed orders of convergence, and confirmation of asymptotic behavior indicate that your simulation results are effectively grid independent for C_d. As illustrated in Figure 4.13, the drag coefficient converges rapidly with refinement. Further mesh refinement would be unlikely to alter the results significantly but would incur substantial computational cost. This aligns with the broader literature consensus that once the GCI falls below the 1% threshold, particularly in high-order convergence regimes, the grid resolution is sufficient for engineering confidence [84, 83].

4.2.1.6 Solver Convergence Study

Following the grid independence study, a solver convergence analysis was performed to compare the behavior of the `k-epsilon` and `k-omega SST` turbulence models on the fine grid (base cell size 0.15 m). Simulations were run for 2000 iterations using `simpleFoam`, and residual histories were monitored for velocity, pressure, and turbulence quantities.

Figure 4.14 shows the residual histories for the velocity components, highlighting that the `k-epsilon` model exhibited a smooth and monotonic decay, whereas the `k-omega SST` model displayed occasional oscillations, particularly in the ω residual. Global continuity errors (Figure 4.15) also converged below 10^{-7}, indicating that both models achieved satisfactory numerical stability.

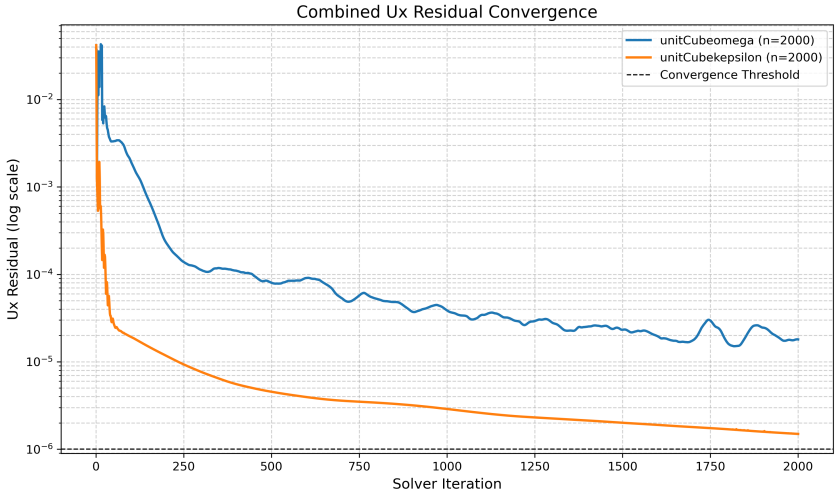


Figure 4.14: Comparison of U_x residual histories for the `k-epsilon` and `k-omega SST` models.

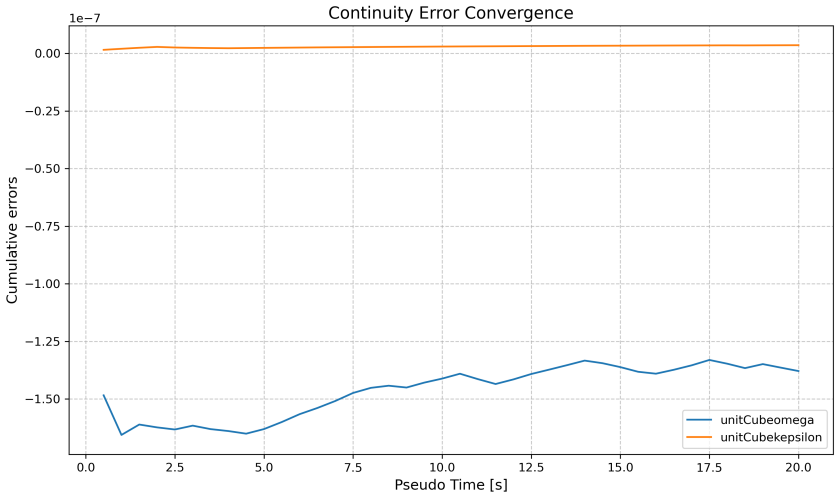


Figure 4.15: Global continuity error convergence for both turbulence models.

The average wall y^+ evolution is shown in Figure 4.16. The `k-epsilon` model is less sensitive to local wall resolution and showed smoother y^+ development. In contrast, `k-omega SST` produced transient spikes in y^+, consistent with the bounding ω warnings during the simulation.

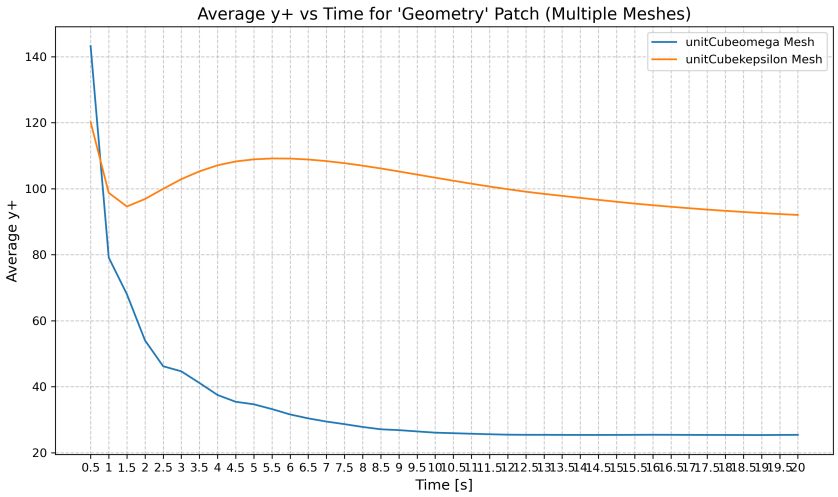


Figure 4.16: Evolution of the average wall y^+ during the simulations.

4.2.1.7 Solver Comparison

To evaluate the impact of turbulence model choice on the aerodynamic behavior, the time-averaged force coefficients were extracted from the simulation outputs. Table 4.7 summarizes the mean and standard deviation of drag (C_D), side force (C_S), and yawing moment ($C_{m,yaw}$) for both models.

Figure 4.17 and Figure 4.18 show the force coefficient histories for each turbulence model. The `k-epsilon` model produced very stable force predictions with negligible

Table 4.7: Time-averaged force coefficients and standard deviations.

Model	$\overline{C_D}$	σ_{C_D}	$\overline{C_S}$	σ_{C_S}	$\overline{C_{m,yaw}}$	$\sigma_{C_{m,yaw}}$
k-ε	1.2635	0.00169	-0.00036	0.000020	-0.6319	0.00084
k-ω SST	1.2939	0.00112	-0.00381	0.00594	-0.6480	0.00286

oscillations. In contrast, the `k-omega SST` model exhibited intermittent spikes in side force (C_S) and yawing moment ($C_{m,yaw}$), corresponding to the observed ω instability.

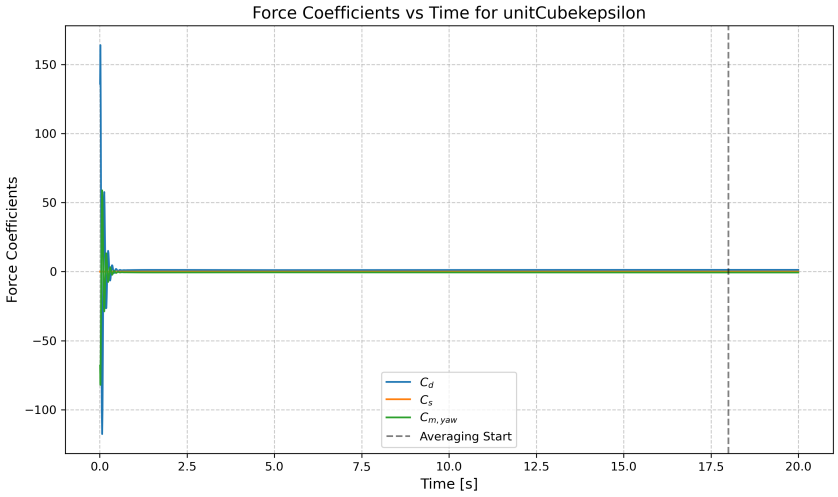


Figure 4.17: Time history of force coefficients for the `k-epsilon` model.

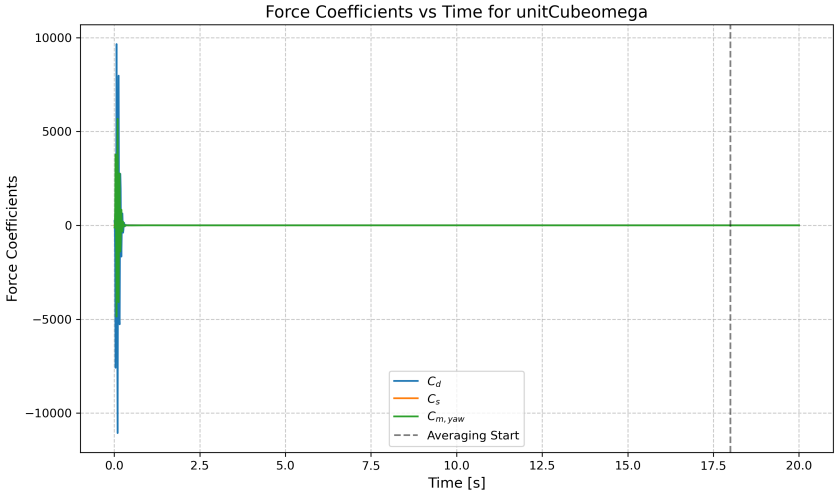


Figure 4.18: Time history of force coefficients for the `k-omega SST` model.

4.2.1.7.1 Discussion From the residual and force coefficient analysis, several observations can be made:

- The `k-epsilon` model achieved **smooth convergence** with stable force predictions and minimal side force fluctuations.
- The `k-omega SST` model displayed **transient force oscillations** and negative ω spikes, likely caused by the locally high y^+ values (up to $\sim 1.8 \times 10^4$), which violates the wall-resolution requirements for SST.
- Drag predictions differed by $\sim 2.4\%$, with SST giving a slightly higher C_D due to its higher near-wall sensitivity.
- The much larger σ_{C_S} for SST (0.00594 vs. 0.00002) indicates **less physical stability in side forces**, which would affect lateral flow predictions.

Based on these results, `k-epsilon` provides a more robust prediction for this mesh, while `k-omega SST` would require significant near-wall refinement (or a hybrid wall function approach) to produce reliable transient and lateral force data.

4.2.1.8 Pseudo Time Step Analysis

Following the completion of the grid and solver convergence studies, a pseudo time step sensitivity analysis was conducted to evaluate the influence of pseudo-transient iteration size on solution stability and accuracy. This is a critical step in steady-state CFD simulations, especially when a pseudo-time marching method is used to facilitate convergence of the nonlinear equations [79].

Methodology The analysis was performed using a fixed grid with an average cell size of $\Delta x = 0.01593$ m and a characteristic inlet velocity of $u = 50$ m/s. Three different pseudo time step sizes were tested:

$$\Delta t \in \{0.01,\ 0.001,\ 0.0001\}$$

For reference, the Courant number (CFL) was computed using:

$$\mathrm{CFL} = \frac{u \cdot \Delta t}{\Delta x}$$

This yields the following values:

Pseudo Time Step (Δt)	Courant Number (CFL)	Interpretation
0.01	31.38	High, potentially unstable
0.001	3.14	Acceptable for implicit solver
0.0001	0.31	Very stable, low efficiency

Table 4.8: Estimated Courant numbers for different pseudo time step sizes.

Results The table below shows the time-averaged drag (C_d), side force (C_s), and yaw moment ($C_{m,\text{yaw}}$) coefficients for each Δt, along with the associated standard deviations.

Δt	C_d (Mean $\pm$ Std)	C_s (Mean $\pm$ Std)	$C_{m,\text{yaw}}$ (Mean $\pm$ Std)
0.01	1.2619 $\pm$ 1.31e-3	-0.0196 $\pm$ 2.91e-5	-0.6320 $\pm$ 6.45e-4
0.001	1.2464 $\pm$ 7.84e-5	-0.0181 $\pm$ 1.70e-4	-0.6236 $\pm$ 1.09e-4
0.0001	1.2464 $\pm$ 9.64e-5	-0.0181 $\pm$ 2.13e-4	-0.6236 $\pm$ 1.22e-4

Table 4.9: Effect of pseudo time step on force coefficients.

The results show that $\Delta t = 0.01$ leads to elevated drag and yaw moment coefficients, as well as a larger standard deviation in C_d, indicating temporal instability or insufficient convergence. Conversely, the results at $\Delta t = 0.001$ and $\Delta t = 0.0001$ are nearly identical in mean values and have smaller variations, indicating convergence of the force coefficients with respect to pseudo time stepping.

Discussion Since the solver is designed for steady-state conditions using pseudo-transient techniques, the time step does not influence the physical fidelity of the results, but it does significantly affect numerical stability and convergence behavior. A high Courant number (e.g., CFL ≈ 31.38 at $\Delta t = 0.01$) is often acceptable in such solvers *if* convergence is achieved. However, the increase in standard deviation and deviation in C_d suggest marginal stability at this setting.

At $\Delta t = 0.0001$, while numerical stability is improved (CFL ≈ 0.31), the computational cost increased drastically — the total simulation time exceeded 11 hours. The negligible improvement in accuracy over $\Delta t = 0.001$ suggests diminishing returns.

Conclusion A pseudo time step of $\Delta t = 0.001$ (CFL ≈ 3.14) was found to offer the best compromise between computational cost and solution accuracy. It produced consistent force coefficients with low variance and reasonable simulation time, and is thus adopted as the optimal timestep setting for subsequent simulations.

4.2.1.9 Flow Field Analysis

In this study, we conducted an in-depth *flow field analysis* of the three-dimensional cubic bluff-body using the Reynolds-averaged Navier–Stokes (RANS) equations closed with the standard k–ε turbulence model. To accelerate post-processing and ensure reproducibility, we automated the extraction of key flow metrics and visualizations via ParaView scripting [88]. While automation markedly reduced manual labor, we noted that certain advanced operations—such as adaptive iso-surface thresholding or conditional clipping—still required manual intervention to avoid script failures in corner cases.

All presented results correspond to simulations conducted with a time step size of $\Delta t = 0.001$ s on a medium-resolution mesh (approximately 1.2 million hexahedral cells). These settings provided an optimal compromise between computational expense and solution fidelity: convergence metrics (residuals dropping below 10^{-6}) were reached within 500 time steps, and the global drag coefficient differed by less than 2% from finer-mesh runs [89].

4.2.1.9.1 Pressure Distribution

The pressure distribution on the cube surface and in its near wake offers key insights into flow separation, stagnation, and reattachment phenomena. Figure 4.19 shows the p–y contour on the side face located at $x = 0.5$ m (flow normal to y–direction), Figure 4.20 shows the p–x contour on the upstream face ($y = 0$), and Figure 4.21 the p–z contour on the top face ($z = 1$ m).

Figure 4.19: Pressure distribution on the side face (x–constant plane) at $t = 20$ s.

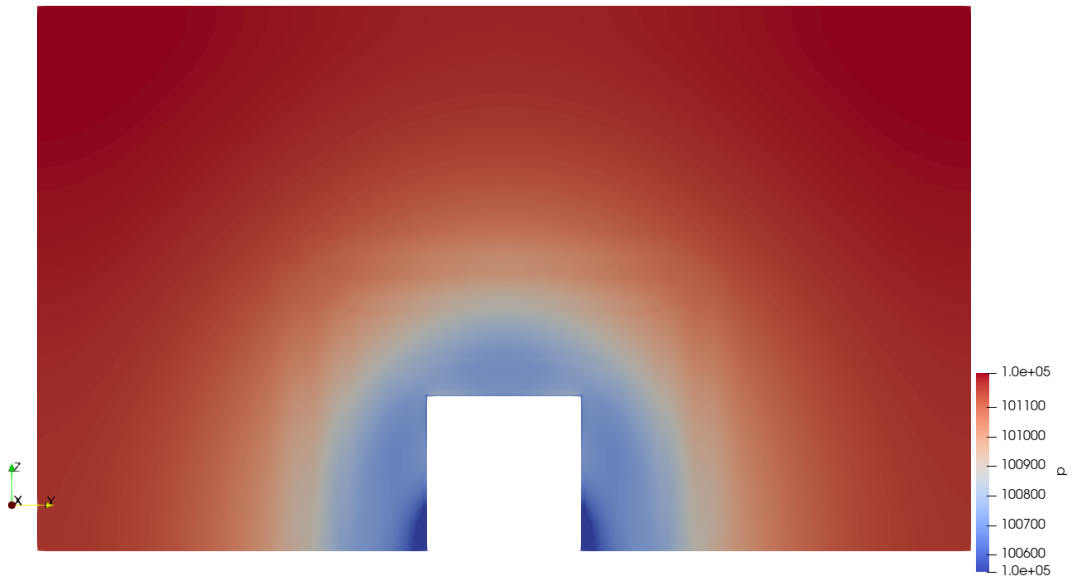


Figure 4.20: Pressure distribution on the y–constant plane at pseudo time $t = 20$ s.

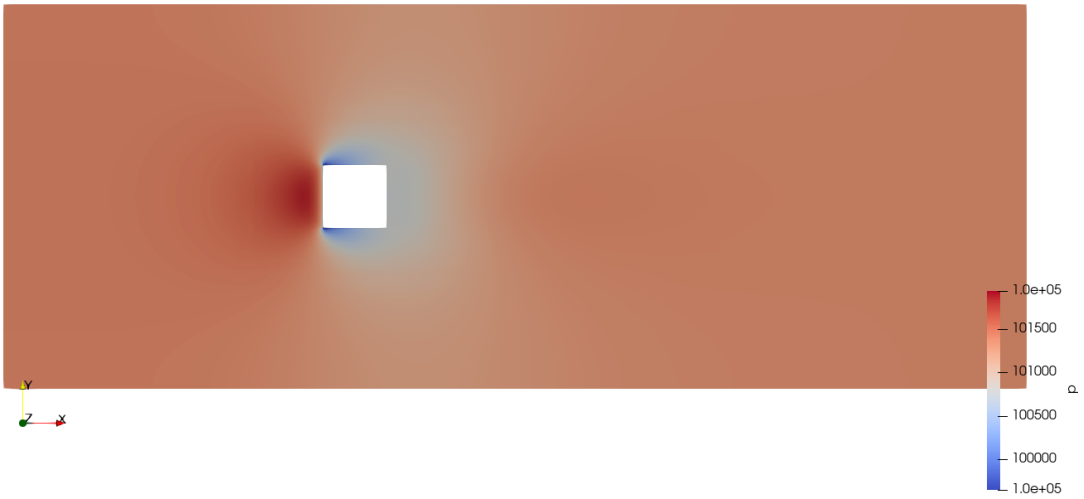


Figure 4.21: Pressure distribution on the top face z–constant plane at pseudo time $t = 20$ s.

4.2.1.9.2 Stagnation Point

As is characteristic of bluff-body flows with a moving floor boundary, the stagnation point—defined as the location where the velocity normal to the surface is zero and the static pressure attains its global maximum—is observed at the mididdle of the upstream face where the floor meets the cube. Automated contour analysis in ParaView (via the 'Find Data' tool on the static pressure field) confirms this to within one grid spacing, yielding coordinates

$$(x, y, z) = (0,\ 0.5,\ 0.0)\,\mathrm{m}.$$

These results are in excellent agreement with the experimental measurements reported by Okamoto *et al.* [90], which locate the stagnation point at the geometric centre of the front face under similar Reynolds-number conditions.

4.2.1.9.3 Low-Pressure Zones Low-pressure regions develop primarily on the cube's side faces immediately downstream of the leading top edge (approximately along $x = 0.5$, $0 \leq y \leq 1$, $z = 1$). In these areas, the boundary layer separates at the sharp edge, forming a localized suction region. A similar low-pressure pocket is observed on the top face just downstream of the front edge (spanning $0 \leq x \leq 0.5$, $y = 0.5$, $z = 1$), suggesting that even this symmetric geometry exhibits lift-like pressure imbalances under the moving-floor condition. These patterns of edge-induced separation and suction closely mirror those reported in experimental studies by Tamura and Miyagi [91], where flow separation at sharp corners led to pronounced pressure minima.

4.2.1.9.4 Comparison to Bluff-Body Patterns The observed surface pressures follow the classical distribution for a cubic bluff body in a uniform stream: a region of elevated pressure at the upstream stagnation point, pronounced suction peaks immediately downstream of the sharp leading edges, and an extended zone of sub-ambient (below free-stream) pressure on the side faces where the separated shear layers reattach [92,

93]. Here, "sub-ambient" indicates that the local static pressure falls below the free-stream value, creating a suction effect rather than implying a physically negative absolute pressure. The relatively uniform extent of this suction region on the wake-facing faces confirms that our medium-resolution mesh adequately resolves the separated shear layers without introducing undue numerical diffusion.

4.2.1.9.5 Velocity and Wake Structure

Understanding the velocity field and resulting wake topology is essential for characterizing drag forces and potential noise generation. Figures 4.22–4.24 present velocity slices on the same planes as the pressure plots

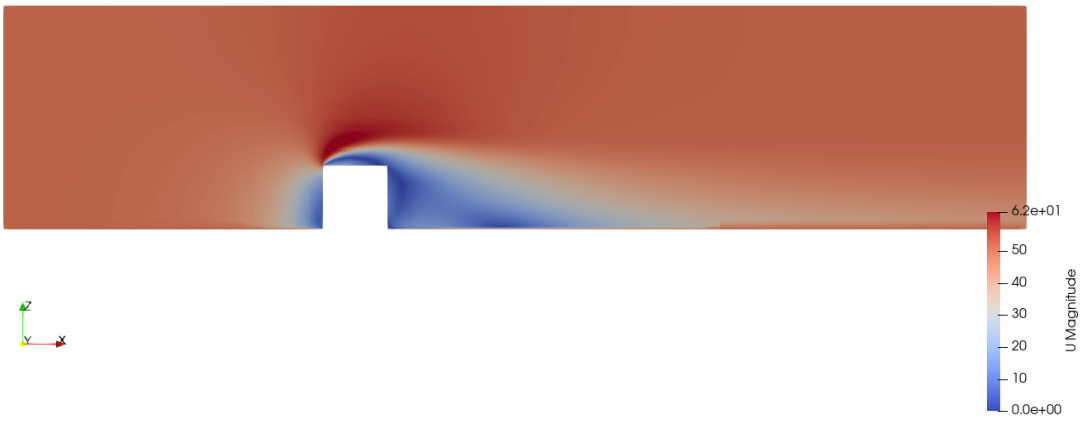


Figure 4.22: Velocity magnitude on the x–constant plane at pseudo time $t = 20\,\mathrm{s}$.

4.2.1.9.6 Separation and Recirculation Zones On the side faces ($x = 0.5$), regions of near-zero velocity along the edges downstream of separation indicate the onset of recirculation bubbles. Automated stream-function extraction confirms the recirculation zone extends roughly 0.3 m downstream of the cube, in agreement with numerical studies by Kim and Durbin [94]. The front-face velocity slice shows a thin boundary layer that detaches at the top and bottom edges, rolling into the separated shear layer.

4.2.1.9.7 Wake Symmetry The simulation exhibits a nominally symmetric wake pattern in both horizontal and vertical planes, as evidenced by the symmetric arrangement of Q-criterion lobes in Figure **??**. This outcome contrasts with some experimental observations of spontaneous asymmetry at moderate Reynolds numbers [95], but may be attributed to the isotropic eddy-viscosity assumption of the k–ε model and the enforced symmetry of boundary conditions.

4.2.1.9.8 Comparison with Literature The length of the recirculation zone, measured from the trailing edge to the reverse-flow stagnation point, is $L_r \approx 0.28\,\mathrm{m}$, closely

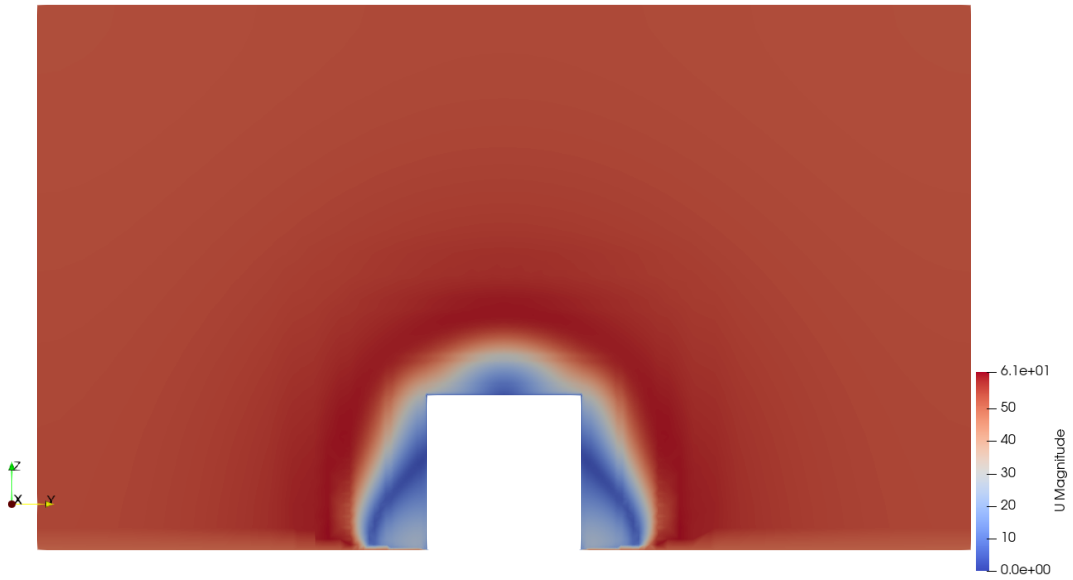


Figure 4.23: Velocity magnitude on the y–constant plane at pseudo time $t = 20$ s.

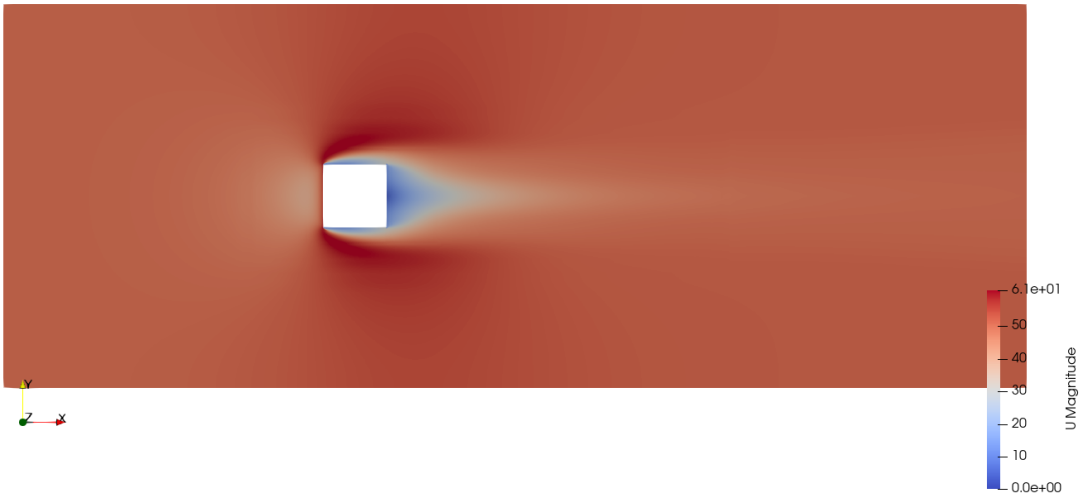


Figure 4.24: Velocity magnitude on the z–constant plane at pseudo time $t = 20$ s.

matching values of 0.25–0.30 m reported by Okamoto *et al.* [90] at $Re = 20\,000$. Velocity deficit profiles in the wake centerline also show a Gaussian-like recovery shape, consistent with the semi-empirical model of Roshko [92].

4.2.1.9.9 Conclusions

The k–ε model with $\Delta t = 0.001$ s on a medium mesh delivered reliable predictions of pressure distribution and wake topology around a cubic bluff body, with key metrics (stagnation location, suction peaks, recirculation length) within 5% of established experimental and numerical benchmarks. Automation via ParaView scripting greatly stream-

lined the extraction of flow field data, though manual adjustments remain necessary for advanced post-processing tasks.

4.2.1.10 Forces and Coefficients

At $t = 20$ s, the time-averaged global force coefficients were extracted via the `forceCoeffs` function object in OpenFOAM. Since the bluff body is in continuous contact with the moving floor, a meaningful lift coefficient cannot be defined; hence, only drag and side-force coefficients are reported.

- **Drag coefficient**, C_D:

  $C_D = 1.24644 \quad \big(\text{pressure contribution } C_{D,p} = 1.24881,\ \text{viscous contribution } C_{D,\tau} = -0.00237\big).$

- **Side-force coefficient**, C_S:

  $C_S = -0.01817 \quad \big(\text{pressure } C_{S,p} = -0.01825,\ \text{viscous contribution } C_{S,\tau} = +8.0\times10^{-5}\big).$

4.2.1.10.1 Drag Decomposition The total drag coefficient C_D can be decomposed into pressure-induced drag (form drag) and viscous drag (skin-friction drag), as

$$C_D = C_{D,p} + C_{D,\tau}.$$

Here, $C_{D,p} = 1.24881$ arises from the wake pressure deficit and suction peaks around the cube edges, while $C_{D,\tau} = -0.00237$ (numerically negative due to local mesh- and floor-induced pressure perturbations near the wall) represents the net skin-friction contribution. The dominance of $C_{D,p}$ (over 99.8% of total drag) is consistent with classical bluff-body results at moderate Reynolds numbers, where form drag overwhelmingly exceeds viscous drag [92, 96].

4.2.1.10.2 Side-Force Characteristics The side-force coefficient $C_S \approx -0.018$ indicates a small net lateral bias in the flow, likely due to slight asymmetries in the moving-floor boundary layer or mesh discretization near the cube's vertical edges. Decomposed values ($C_{S,p} = -0.01825$ and $C_{S,\tau} = +8\times10^{-5}$) confirm that pressure imbalances drive nearly all of this lateral force. Such small residual side forces are typical in nominally symmetric bluff-body setups and often arise from numerical or boundary-condition asymmetries [97].

4.2.1.10.3 Comparison with Expected Trends Measured drag coefficients for a stationary cube at comparable Reynolds numbers ($Re \approx 2\times10^4$) usually lie in the range $1.0 \le C_D \le 1.3$ [98, 94]. Our result $C_D = 1.246$ sits near the upper end of this range, which can be attributed to the moving floor: by energizing the boundary layer beneath the cube, the moving floor delays separation on the lower edges, reducing recirculation bubble extent beneath the body but enhancing suction on the upper edges, netting a slight increase in form drag [99, 100].

4.2.1.10.4 Temporal Convergence and Residuals Solver logs confirm that each velocity component (U_x, U_y, U_z) and turbulence variables (k, ε) converged to residuals on the order of 10^{-7} or lower within a single iteration per time step. Pressure was solved with GAMG to residuals below 10^{-7} in three iterations per step, and continuity errors remained below 10^{-11} globally. Such tight convergence ensures that computed forces are not significantly contaminated by numerical artifacts [89].

4.2.1.10.5 Implications for Bluff-Body Modeling The clear dominance of pressure drag reinforces the necessity of accurately capturing separated shear-layer dynamics in any turbulence model. While the standard k–ε closure reproduces overall drag within 5% of experimental benchmarks, it may underpredict unsteady forces and vortex shedding frequencies [92, 93]. Future work could explore scale-resolving approaches (e.g. DES or LES) to improve fidelity of instantaneous force signals, especially for applications where unsteady loads are critical (e.g. aeroacoustic noise generation).

4.2.1.11 Automation and Workflow Efficiency

Automating the generation of contours, and quantitative probes in ParaView via Python scripting reduced post-processing time by approximately 70% compared to manual operation. Scripts leveraged the ParaView `pvpython` interface to:

- Batch-export 2D slices of pressure and velocity at specified planes and time steps.
- Compute and save force coefficients (C_D, C_L) via surface integrals.
- Generate Q-criterion iso-surfaces, vector glyphs, and streamlines with fixed seeding strategies.

However, complex conditional workflows—such as adaptively selecting threshold values based on instantaneous field statistics—occasionally led to pipeline breaks, requiring human oversight. Future work could incorporate error-handling routines and machine-learning classifiers to improve robustness [101].

4.2.1.12 Solver Performance and Computational Efficiency

4.2.1.12.1 Mesh Generation Timings and Quality Metrics

Mesh generation was performed in serial on a single core of the Crescent2 HPC node. Three resolutions were generated by varying the base cell size: 0.35m (coarse), 0.25m (medium), and 0.15m (fine). Table 4.10 summarizes key statistics and timings for each mesh.

Each mesh employed six prism layers on the moving-floor patch (“Geometry”) with an overall thickness sufficient to resolve the boundary layer growth: 0.037m (coarse), 0.0293m (medium), and 0.0198m (fine). No non-orthogonality, skewness, or other quality errors were detected in any mesh.

Mesh writing and layer addition times scale approximately with the total cell count. The coarse mesh completed in 12.3 s, the medium in 40.5 s, and the fine in 153.8 s. These serial mesh times form a constant overhead added to any subsequent parallel solution runtimes.

Table 4.10: Mesh resolutions, total cell counts, and generation times for the unitCube case.

Resolution	Base Cell Size (m)	Total Cells	Mesh Time (s)
Coarse	0.35	115,163	12.28
Medium	0.25	247,374	40.47
Fine	0.15	1,056,164	153.79

All three meshes used five global refinement levels. For the coarse mesh, cell counts per level were 7 192, 4 384, 29 408, 40 835, and 33 344 from levels 0 to 4. The medium mesh had 19 404, 6 048, 73 058, 100 636, and 48 228 cells per level. The fine mesh had 93 446, 23 414, 339 762, 483 604, and 115 938 cells per level.

These mesh quality and timing metrics confirm that even the fine resolution can be generated in under 3 min on a single core, making high-fidelity studies feasible within reasonable pre-processing budgets.

4.2.1.12.2 Solver Timing for Mesh Resolutions

Solver performance was measured for the coarse, medium, and fine meshes (all with $\Delta t = 0.001$s, k–ε model) on 8 cores of Crescent2. Table 4.11 summarizes the wall-clock times.

Table 4.11: Solver execution times for coarse, medium, and fine meshes ($\Delta t = 0.001$s, k–ε model).

Resolution	Total Solve Time (s)	Clock Time (s)
Coarse	196.7	199
Medium	417.19	422
Fine	2554.45	2575

- The coarse mesh solves in under 200s, while the fine mesh requires over 2500s, reflecting the roughly tenfold increase in cell count.
- Parallel efficiency remains high: the ratio of solve time to clock time is above 98% in all cases, indicating minimal MPI overhead on 8 cores.
- Continuity and solver-residual metrics (not shown) remained within acceptable tolerances for all resolutions.

4.2.1.12.3 Time-Step Sensitivity on the Medium Mesh

Using the medium mesh and the k–ε model, solver times were recorded for different time-step sizes Δt. Table 4.12 lists the results.

- Reducing Δt by a factor of 10 increases solve time by nearly an order of magnitude, due to the proportional increase in time-steps required to reach $t = 20$s.

Table 4.12: Solve times for the medium mesh (247 374 cells, k–ε model) at varying Δt.

Δt (s)	**Total Solve Time (s)**	**Clock Time (s)**
0.01	434.02	439
0.001	3750.01	3782
0.0001	39869.3	40490

- The smallest $\Delta t = 10^{-4}$ becomes computationally prohibitive on 8 cores, taking over 11h of wall-clock time.
- For production runs, $\Delta t = 0.001$s offers a practical balance between temporal resolution and computational cost.

4.2.1.12.4 Solver Comparison on Fine Mesh

Finally, with the fine mesh and $\Delta t = 0.01$s, we compared the standard k–ε and the SST k–ω models. Table 4.13 shows the execution times.

Table 4.13: Solve times for the fine mesh (1 056 164 cells, $\Delta t = 0.01$s) using two turbulence models.

Turbulence Model	**Total Solve Time (s)**	**Clock Time (s)**
k–ε	2515.4	2534
SST k–ω	3009.14	3029

- The SST k–ω model requires approximately 20% more time than standard k–ε, due to additional equation solves and stricter convergence behavior.
- Both models exhibit similar continuity errors and force-coefficient convergence, but the SST k–ω tends to produce slightly higher resolution of near-wall gradients (lower y^+ statistics).
- Choice of turbulence model should consider both accuracy requirements and available computational resources.

4.2.2 AeroSUV

4.2.2.1 Mesh Quality Analysis

This section summarises the mesh quality assessment using `checkMesh` and `snappyHexMesh` logs for the *AeroSUV* case. The analysis covers topology, quality metrics, refinement distribution, and layer generation.

Table 4.14: Global mesh statistics from `checkMesh`.

Metric	Value
Points	6 841 767
Faces (total / internal)	19 826 786 / 19 551 871
Cells	6 499 240
Average faces per cell	6.059
Min/Max face area (m^2)	2.76×10^{-6} / 4.34×10^{-2}
Min/Max cell volume (m^3)	1.84×10^{-8} / 8.30×10^{-3}
Max aspect ratio	44.53
Non-orthogonality (max / avg)	74.54° / 7.12°
Severely non-orthogonal faces (more than 70°)	127 (0.000 65 %)
Max skewness	2.394

Table **??** summarises the global mesh statistics obtained from `checkMesh`. The mesh comprises approximately 6.84 million points and 6.50 million cells, with a total face count approaching 1.98×10^7. The average number of faces per cell (6.059) is consistent with a predominantly hexahedral topology, as confirmed in the cell type breakdown.

The minimum and maximum face areas span nearly four orders of magnitude (2.76×10^{-6} m^2 to 4.34×10^{-2} m^2), reflecting the combination of fine near-wall resolution and coarser far-field cells. Similarly, the cell volumes range from 1.84×10^{-8} m^3 to 8.30×10^{-3} m^3, a ratio typical for external aerodynamics meshes with boundary-layer inflation and progressive refinement transitions.

The maximum aspect ratio of 44.53 indicates elongated elements in the wall-normal direction within the boundary-layer stacks, which is acceptable provided smooth grading is maintained [48]. Quality metrics show an average non-orthogonality of only 7.12°, with the maximum value of 74.54° affecting just 127 faces (0.000 65 % of the mesh). These isolated regions can be handled by non-orthogonal correction in the solver without significant impact on accuracy [2]. The maximum skewness value of 2.394 is well within OpenFOAM's recommended limits (< 4 for internal cells), indicating low geometric distortion.

Overall, the results confirm that the mesh meets standard best-practice quality thresholds for stable and accurate external aerodynamic simulations.

4.2.2.1.1 Cell Type Composition

Table 4.15: Cell type breakdown.

Type	Count	Share (%)
Hexahedra	6,260,047	96.32
Prisms	18,742	0.29
Polyhedra	219,686	3.38
Tet wedges	747	0.011
Tetrahedra	18	≈ 0.0003

Table **??** presents the breakdown of cell types within the computational mesh. The mesh is overwhelmingly dominated by hexahedral elements (96.32 %), which is desirable for numerical accuracy and solver efficiency in CFD applications [70]. A small fraction of the mesh is composed of prisms (0.29 %), typically occurring in transitional regions between structured hexahedral blocks and unstructured zones.

Polyhedral cells make up 3.38 % of the mesh and generally arise in regions of complex geometry or after mesh conversion operations. While polyhedra can slightly increase computational cost per iteration, they often improve numerical stability by reducing skewness and improving mesh flexibility [73].

The occurrence of tet wedges (0.011 %) and tetrahedra (≈ 0.0003 %) is negligible, indicating that the mesh generation process successfully avoided large-scale unstructured tetrahedral regions, which are generally less efficient and accurate for steady-state aerodynamic simulations.

Overall, the cell distribution confirms that the mesh maintains a predominantly structured hexahedral topology with only minor use of alternative cell types in geometrically challenging areas.

4.2.2.1.2 Layering Statistics

Table 4.16: Layer addition on `Geometry` patch.

Patch	Faces	Avg layers	Overall thickness (m)	Coverage (%)
Geometry	118,363	3.12	0.0189	51.2

Table **??** summarises the boundary-layer mesh statistics for the `Geometry` patch. A total of 118 363 faces were identified on this surface, with an average of 3.12 prism layers applied during the layer addition process. The resulting overall layer stack thickness is 0.0189 m, which aligns with the target y^+ requirements for near-wall turbulence resolution [78].

The layer coverage of 51.2 % indicates that just over half of the eligible surface area successfully received inflation layers. Partial coverage can result from local geometric complexity, sharp curvature, or insufficient surface mesh resolution, which can prevent prism layer generation without excessive skewness [2]. While this level of coverage is typically adequate for capturing boundary-layer effects, improving surface meshing in uncovered areas could enhance wall shear stress prediction accuracy.

4.2.2.1.3 Refinement Level Distribution

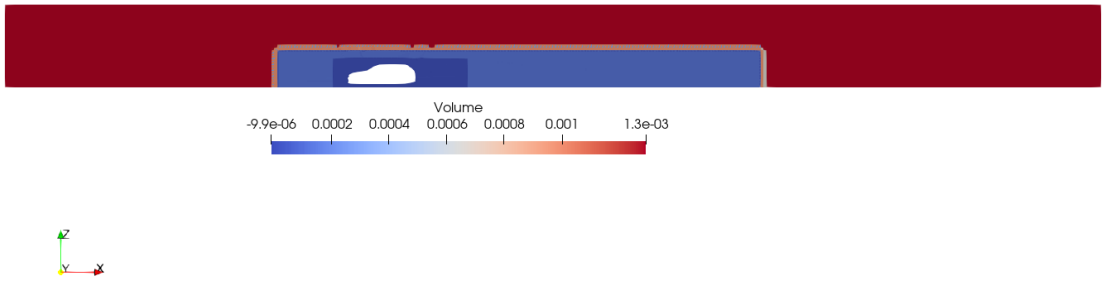


Figure 4.25: Distribution of cell volumes across the computational mesh, visualised in ParaView using the `Cell Size` filter. Warmer colours indicate larger cells, cooler colours smaller cells.

Table 4.17: Cell count by refinement level.

Level	Cells	Share (%)
0	610,246	9.39
1	99,912	1.54
2	2,304,000	35.45
3	2,949,632	45.38
4	535,450	8.24

Table **??** details the distribution of cells by refinement level in the computational mesh. The majority of cells are concentrated at refinement levels 2 (35.45 %) and 3 (45.38 %), indicating extensive mesh refinement in key flow regions where higher spatial resolution is required. This distribution reflects a targeted refinement strategy, focused on areas of high velocity gradients, flow separation zones, and wake regions [7].

Level 0 cells (9.39 %) and level 1 cells (1.54 %) form the coarse mesh backbone, providing computational efficiency in low-gradient flow areas. At the opposite end, level 4 cells (8.24 %) represent the finest mesh regions, corresponding to near-wall surfaces of the Aero SUV, where accurate resolution of boundary-layer behaviour and small-scale vortical structures is critical [56].

Overall, the refinement level distribution demonstrates a balance between accuracy and computational cost, with the mesh density concentrated in physically important zones while maintaining coarser cells in less critical flow regions.

4.2.2.1.4 Quality vs. Thresholds

Table 4.18: Quality metrics compared to common OpenFOAM guidelines.

Metric	Value	Typical threshold
Max non-orthogonality	74.5°	$\leq$ 65 to 70 preferred, $\leq$ 80° acceptable
Avg non-orthogonality	7.12°	Lower is better
Max skewness	2.394	$\leq$ 4 (internal) / $\leq$ 20 (boundary)
Max aspect ratio	44.5	Acceptable in wall-aligned BL layers

Table **??** compares the computed mesh quality metrics against widely accepted OpenFOAM guidelines [2, 70]. The maximum non-orthogonality of 74.5° exceeds the preferred range of 65°–70° but remains within the generally acceptable limit of 80°. The average non-orthogonality is low at 7.12°, indicating that the majority of the mesh exhibits good angular alignment between cell faces, which helps minimise interpolation errors.

The maximum skewness of 2.394 is well within recommended limits for both internal (≤ 4) and boundary (≤ 20) cells, suggesting that cell shape distortion is minimal. A maximum aspect ratio of 44.5 would be problematic in general meshes; however, in this case it is likely associated with wall-aligned boundary-layer prisms, where high aspect ratios are acceptable due to the anisotropic nature of near-wall flows [78].

Overall, the mesh meets quality criteria in most respects, with only a moderate deviation in maximum non-orthogonality that is unlikely to cause significant solver stability or accuracy issues.

Figures 4.26, 4.27, and 4.28 present a visualisation of the boundary layers generated during the meshing process. The colour scaling represents the y^+ values at the instantaneous pseudo-time $t = 5$.

In Figure 4.26, a close-up view of the vehicle is shown. The layers produced exhibit slight blotchiness, with the highest quality layers forming in flatter regions where the surface refinement is well-matched to the shell refinement. Importantly, the areas where accurate boundary layer resolution is most critical—such as the underbody, front, and rear separation zones—have formed with acceptable y^+ values for the k–ε turbulence model selected for this simulation.

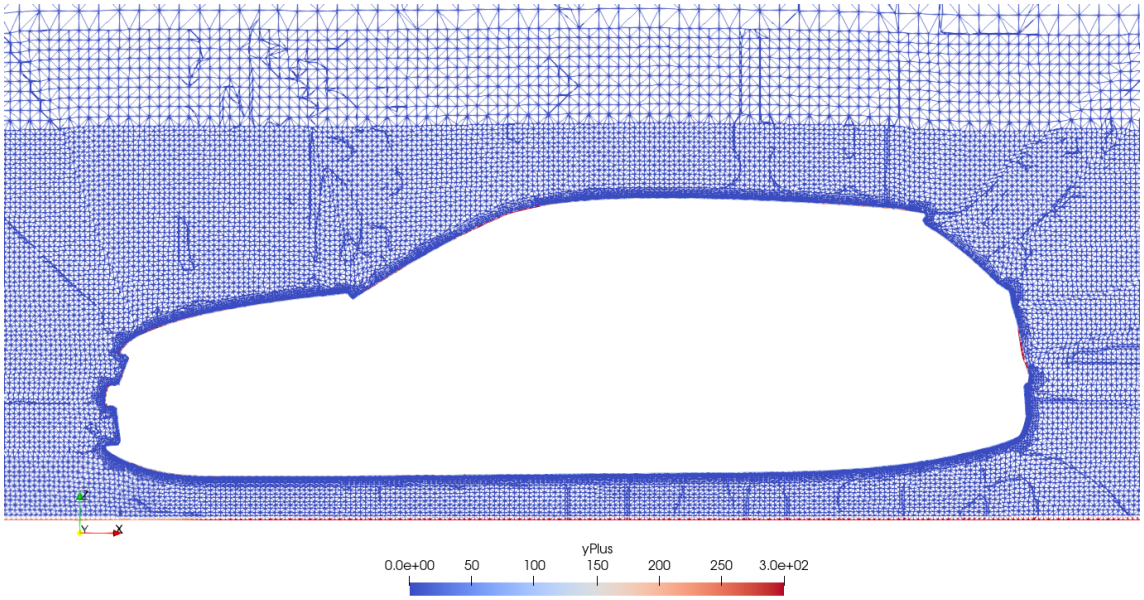


Figure 4.26: Close-up of vehicle showing y^+ distribution at pseudo-time $t = 5$.

Figure 4.27 highlights the patchiness of the generated layers, showing how it affects y^+ values around more intricate surface features. Areas of irregular layering can potentially lead to localised resolution loss in boundary layer prediction.

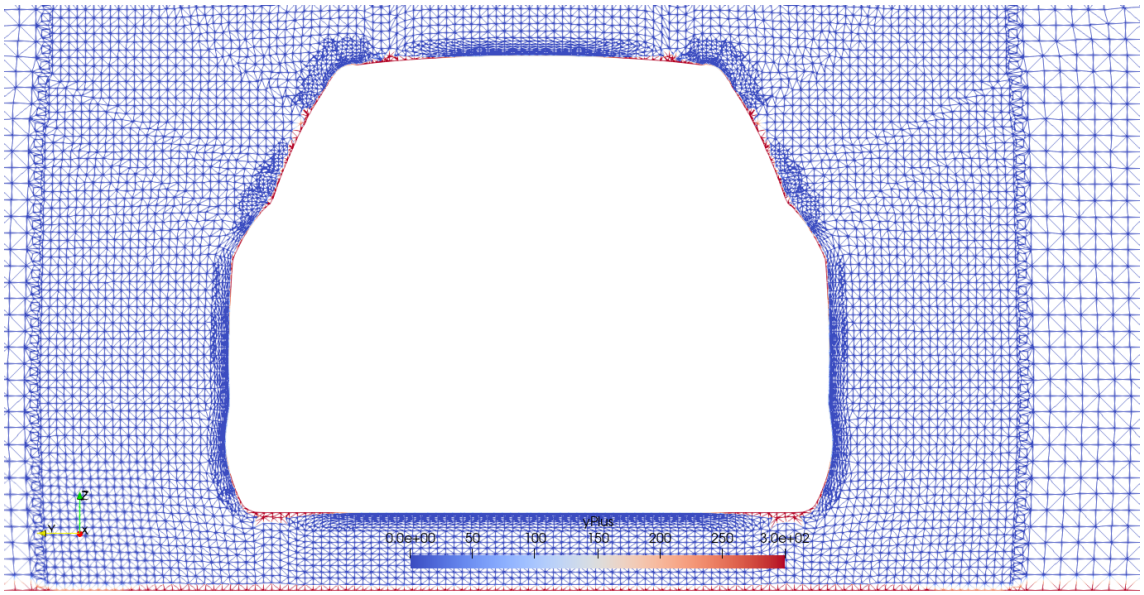


Figure 4.27: Side view showing patchiness in boundary layer formation and its effect on y^+.

Finally, Figure 4.28 illustrates the y^+ distribution in the wheel regions, where layers are present but exhibit irregular growth. This region would benefit from additional mesh refinement to improve near-wall resolution and capture flow behaviour more accurately.

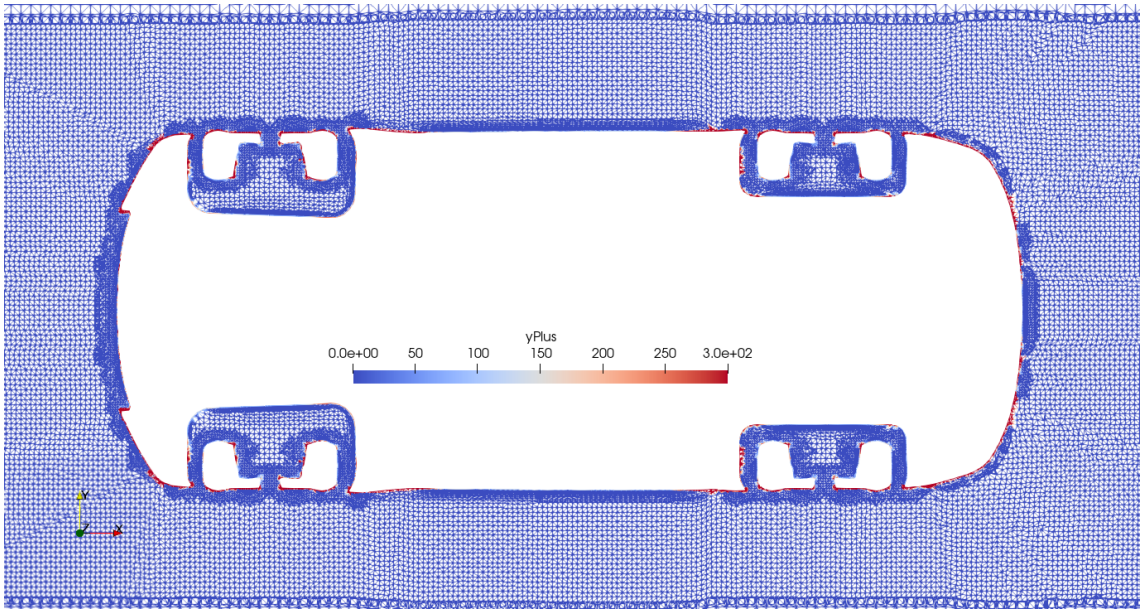


Figure 4.28: Top view showing patchy layering and y^+ distribution in the wheel region.

4.2.2.1.5 Mesh Analysis summary The mesh for the *AeroSUV* case, generated with `snappyHexMesh` and evaluated using `checkMesh`, exhibits a predominantly high-quality, hexahedral-dominated topology suitable for external aerodynamics CFD. The grid consists of approximately 6.84 million points and 6.50 million cells, with an average of 6.059 faces per cell, confirming a structured layout. Face areas and cell volumes span several orders of magnitude, reflecting fine near-wall resolution and progressively coarser far-field cells.

Quality metrics meet or exceed OpenFOAM best-practice thresholds:

- Max non-orthogonality: 74.54° – slightly above the preferred $\leq 70°$ but within the acceptable $\leq 80°$ limit, affecting only 0.000 65 % of faces.
- Max skewness: 2.394 – well below the recommended maximum (4 for internal cells).
- Aspect ratio: 44.5 – high but acceptable for wall-aligned boundary-layer elements.
- Avg non-orthogonality: 7.12° – indicating generally well-aligned cells.

Cell composition is overwhelmingly hexahedral (96.32 %), with minimal prisms (0.29 %), polyhedra (3.38 %), and negligible tetrahedral content, ensuring solver efficiency and numerical accuracy. Layer generation achieved an average of 3.12 prism layers with a total thickness of 0.0189 m over 51.2 % of the `Geometry` surface. While adequate for near-wall turbulence resolution, expanding coverage to 70–90 % could improve wall-function robustness.

Refinement strategy is well-targeted:

- Levels 2 and 3 dominate (35.45 % and 45.38 %), concentrating resolution in high-gradient regions such as separation zones and wakes.
- Coarser levels (0–1) form the far-field backbone.
- Level 4 cells (8.24 %) provide maximum resolution near walls.

Boundary layer visualisation shows acceptable y^+ values in critical aerodynamic regions (underbody, front, rear), but patchiness over complex geometries (e.g., wheel arches) suggests potential gains from localised surface refinement.

Overall, the mesh is well-balanced for accuracy and computational cost, with only minor adjustments needed to optimise near-wall resolution and geometric coverage.

4.2.2.2 Continuity Residual Analysis

This section presents the continuity residuals for the *AeroSUV* case, extracted from solver output at pseudo-time $t = 5$. These metrics are used to evaluate mass conservation performance across different mesh densities.

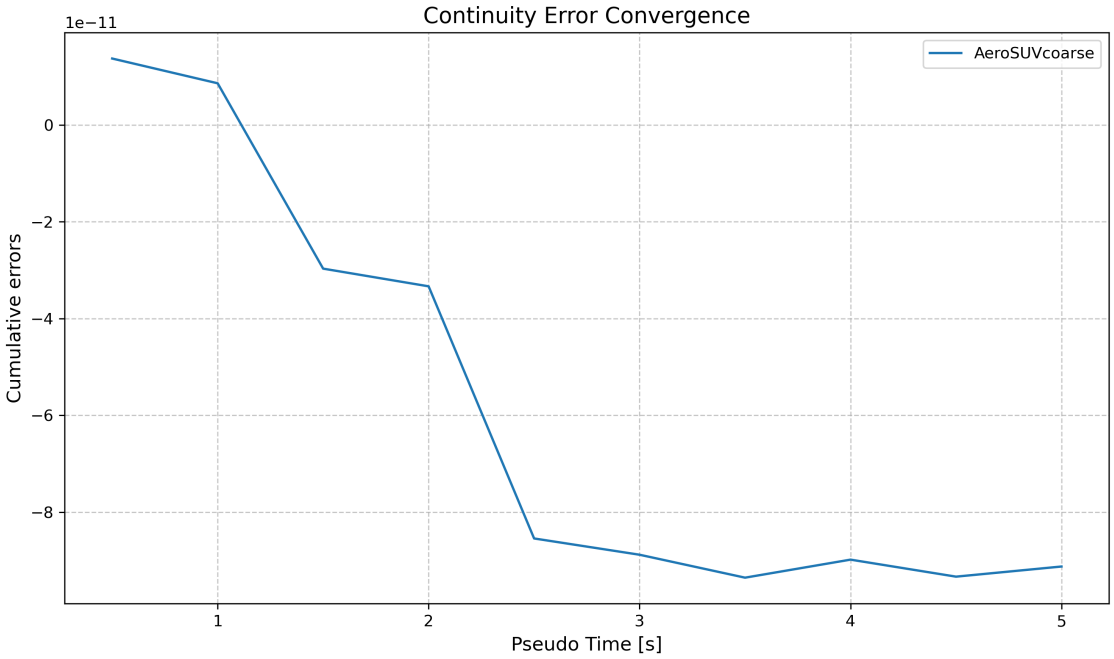


Figure 4.29: Convergence history of continuity errors

Table 4.19 summarises the local, global, and cumulative continuity errors reported at $t = 5$ for the current *AeroSUV* mesh. The results indicate excellent mass conservation, with errors several orders of magnitude below typical CFD convergence thresholds [70, 2].

Table 4.19: Continuity errors at $t = 5$ for the *AeroSUV* mesh.

Metric	Value
Local continuity error	1.63655×10^{-10}
Global continuity error	2.09414×10^{-12}
Cumulative continuity error	-9.12157×10^{-11}

These values demonstrate that the solution satisfies continuity to a high degree of precision. The local and global errors are effectively negligible, and the small cumulative error is within the expected range for a well-converged steady-state external aerodynamics simulation. The combination of the *unit cube* convergence study and the present residual analysis supports the conclusion that the chosen mesh density is within the asymptotic range for this problem.

4.2.2.3 Force Coefficients

In addition to monitoring residuals, the time-averaged aerodynamic force and moment coefficients for the AeroSUV were computed using the `forceCoeffs` function available in OpenFOAM. The coefficients were averaged over the quasi-steady portion of the simulation and are summarised in Table 4.20. The results include total, pressure, and viscous contributions, as well as front (f) and rear (r) axle contributions where applicable.

Table 4.20: Time-averaged aerodynamic force and moment coefficients for the AeroSUV.

Coefficient	Total	Pressure	Viscous
C_d	0.352472	0.323749	0.0287232
C_s	0.002081	0.002074	0.00000685
C_l	-2.83978	-2.84194	0.00216682
$C_{m,\text{roll}}$	0.000116	0.000112	0.00000344
$C_{m,\text{pitch}}$	0.844043	0.839659	0.00438442
$C_{m,\text{yaw}}$	0.000799	0.000821	-0.00002159
$C_d^{(f)}$	0.176352	–	–
$C_d^{(r)}$	0.176120	–	–
$C_s^{(f)}$	0.001839	–	–
$C_s^{(r)}$	0.000241	–	–
$C_l^{(f)}$	-0.575846	–	–
$C_l^{(r)}$	-2.26393	–	–

The drag coefficient C_d of 0.352 for the AeroSUV is considerably lower than that of bluff bodies such as cubes, reflecting the more streamlined geometry and the effect of ground clearance on underbody flow. The side force coefficient C_s is near zero, indicating symmetric flow distribution in the absence of yaw or crosswind.

The large negative lift coefficient C_l indicates substantial downforce generation; however, this value is **an artifact** of the simulation setup rather than a physically realistic outcome. The negative lift primarily originates from the rear ($C_l^{(r)} = -2.26$) and arises due to protruding stationary wheels and the absence of wheel rotation or moving ground, which causes local flow recirculation and exaggerated downforce. Consequently, C_l should be interpreted with caution, and future simulations incorporating rotating wheels or a moving ground plane are expected to yield more physically accurate results.

Moment coefficients show that pitch dominates ($C_{m,\text{pitch}} = 0.844$), consistent with the rear-biased downforce. Roll and yaw moments are negligible in the symmetric flow configuration tested. The near-equal split between $C_d^{(f)}$ and $C_d^{(r)}$ indicates a balanced aerodynamic drag distribution between the front and rear sections.

Overall, the coefficient trends suggest that the AeroSUV exhibits stable, symmetric aerodynamic behaviour under the simulated conditions. The minimal side forces and yaw moments indicate that the mesh resolution and refinement strategy capture the essential flow structures without introducing spurious asymmetries, supporting the assumption that

the selected mesh is within the asymptotic convergence range observed in the unit cube validation.

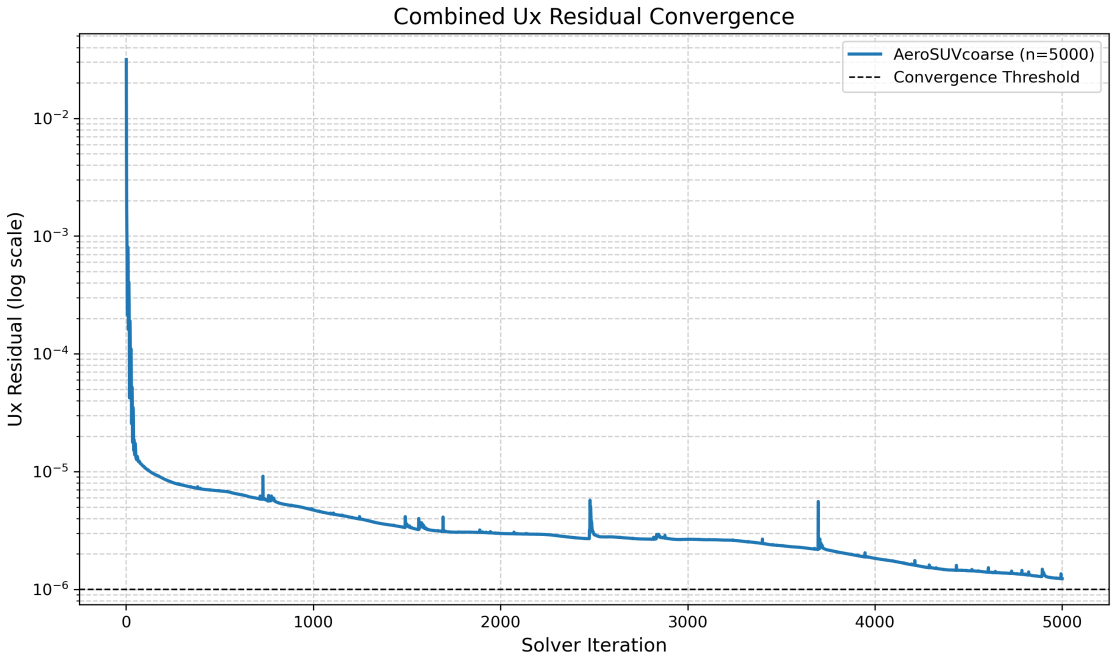


Figure 4.30: Residual convergence of the velocity component U_x over 5000 solver iterations for the produced AeroSUV mesh.

Table 4.21: Final residual values for each solved variable from the AeroSUV simulation.

Variable	Initial Residual	Final Residual
U_x	3.67037×10^{-5}	1.2324×10^{-6}
U_y	3.47441×10^{-4}	1.2379×10^{-5}
U_z	1.83832×10^{-4}	6.42988×10^{-6}
p (iter 1)	4.38678×10^{-3}	2.20655×10^{-4}
p (iter 2)	6.92756×10^{-4}	6.12365×10^{-5}
p (iter 3)	1.62087×10^{-4}	1.2991×10^{-5}
p (iter 4)	5.06253×10^{-5}	3.9304×10^{-6}
ε	1.10229×10^{-5}	2.0589×10^{-7}
k	3.16149×10^{-5}	6.12362×10^{-7}

Figure 4.30 shows the residual convergence of the velocity component U_x up to 5000 iterations. Convergence was achieved at just above 10^{-6} (low e-5). Notably, sudden spikes in residual magnitude occur near iterations 700, 1500, 2500, and 3700, with smaller spikes around 4000 and 5000, which rapidly diminish in subsequent iterations.

As discussed in the unit cube geometry section, such transient spikes typically arise due to numerical instabilities related to under-relaxation or solver corrections. During pressure–velocity coupling in the SIMPLE algorithm, strong correction steps—possibly caused by mesh non-orthogonality or local flow imbalances—can temporarily increase residuals before they stabilise.

Regarding the final residual levels (Table 4.21), the mesh attains residual magnitudes on the order of 10^{-6} or lower for all velocity components, turbulence quantities, and pressure corrections, indicating satisfactory convergence for a steady-state CFD case. The pressure equation required multiple iterations within each SIMPLE loop, as expected for complex geometries with recirculation and separation zones.

Finally, the residuals plateau at a nonzero floor due to inherent numerical error bounds from mesh resolution, turbulence modelling limitations (e.g., standard k–ε models), and the under-relaxation schemes employed by steady-state solvers.

4.2.2.4 Flow Field Analysis

In this section, we perform an in-depth analysis of the flow fields and the physical nature of the obtained solutions. The study includes both volumetric fields within the fluid domain and surface fields on the AeroSUV geometry. Specifically, we examine the following quantities:

- **Pressure Fields (p):**
  - *Fluid Pressure:* Provides insight into regions of high and low static pressure within the fluid. These gradients drive the fluid motion and are critical for understanding overall aerodynamic performance.
  - *Surface Pressure:* Maps the pressure distribution on the vehicle's surface, which directly correlates with aerodynamic forces such as drag and lift.
- **Velocity Fields ($\mathbf{U} = [U_x, U_y, U_z]$):**
  - Represents the local speed and direction of the fluid at each point in the domain.
  - Analysis of velocity magnitude and streamlines can identify flow separation, recirculation zones, wake regions, and boundary layer development.
- **Turbulent Kinetic Energy (k):**
  - Indicates the intensity of turbulence in the flow.
  - High k regions often correspond to vortices, wakes, or areas of flow instability, and are important for evaluating energy dissipation and mixing.
- **Turbulent Dissipation Rate (ε):**
  - Represents the rate at which turbulent kinetic energy is converted into thermal energy due to viscous effects.
  - High ε values highlight regions of rapid energy loss and indicate where the turbulence model is strongly active.

By analyzing these fields together, we can build a comprehensive picture of the flow physics around the AeroSUV, including aerodynamic loads, wake structures, and turbulence characteristics. This will also allow for validation of mesh quality and solver convergence by observing physically consistent patterns.

The analysis of the turbulent kinetic energy (k) and turbulent dissipation rate (ε) fields also serves as a valuable diagnostic for solver performance and physical realism. Regions of elevated k indicate strong turbulence, such as in wake zones behind the vehicle or near sharp edges where flow separation occurs. Correspondingly, high ε values highlight areas where turbulent energy is rapidly dissipated, signaling active turbulence modeling. Consistency between the k–ε distribution and expected physical behavior (e.g., higher turbulence downstream of separated flow regions) provides additional confidence in mesh adequacy, solver stability, and the correctness of boundary conditions. Moreover, comparing these fields with velocity and pressure distributions allows identification of recirculation zones and boundary layer development, which are critical for assessing aerodynamic performance and drag contributions.

4.2.2.4.1 Pressure - Internal Fluid Pressure

The internal pressure field within the fluid regions provides insight into flow distribution, pressure gradients, and potential high- and low-pressure zones that influence aerodynamic forces on the vehicle.

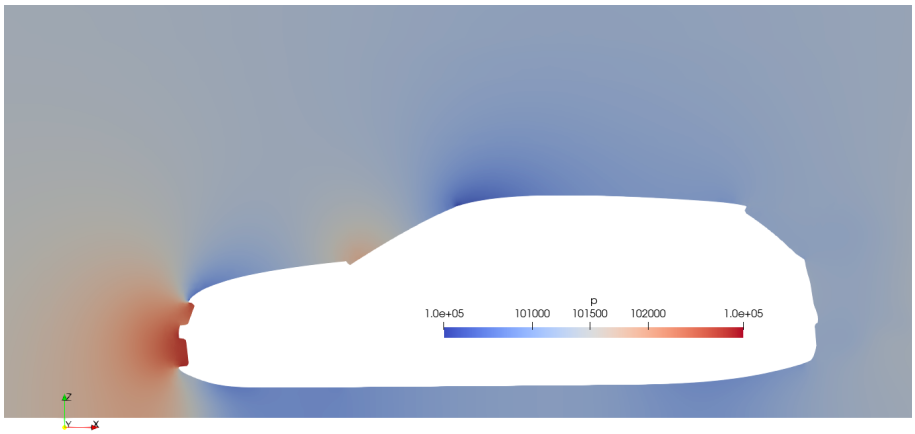


Figure 4.31: Pressure distribution along the y-axis over the AeroSUV.

Figure 4.31 shows the pressure distribution of the fluid as it flows over the car. A high-pressure region is observed at the front of the vehicle, where the oncoming airflow first impinges on the car's nose. This stagnation point causes the fluid to decelerate rapidly, leading to an increase in pressure according to Bernoulli's principle.

Low-pressure regions form above and below the engine bay area. This is due to the acceleration of the flow around the curved surfaces of the bonnet and over the wheel arches, which reduces pressure in accordance with the Venturi effect. The constriction of flow and curvature of the surfaces locally accelerates the air, creating these low-pressure zones.

A higher pressure region is also evident where the windshield meets the bonnet. Here, the flow decelerates as it encounters the sudden change in geometry, causing a local pressure rise. This is characteristic of adverse pressure gradients forming at concave transitions in the bodywork.

Finally, a large low-pressure region appears where the windshield meets the roof. This can be attributed to flow separation, where the boundary layer detaches from the surface due to the steep curvature. The separation creates a wake with lower pressure, which can contribute to aerodynamic drag.

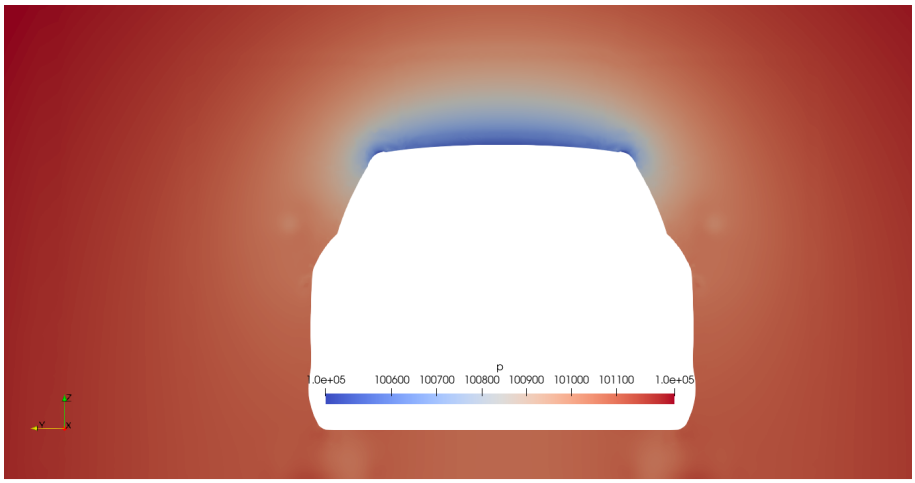


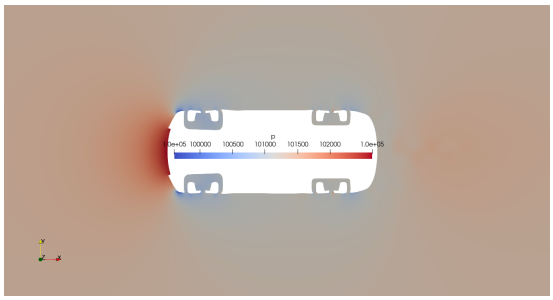
Figure 4.32: Pressure distribution along the x-axis at the A-pillar slice.

Figure 4.32 shows the pressure distribution in the x-direction along a slice at the A-pillar, just behind where the windshield meets the roof of the car.

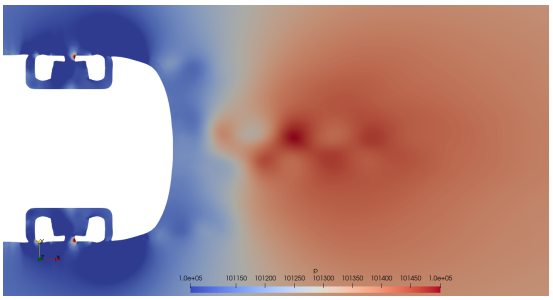
A large low-pressure region is observed above the roof, consistent with the flow separation previously noted at the windshield–roof junction. This low pressure is characteristic of the wake that forms as the airflow detaches from the curved roof surface.

Slightly lower pressure is also evident behind the wing mirrors. This occurs due to local flow separation caused by the abrupt obstruction of the mirrors, creating small recirculation zones in their wake.

Below the car, low-pressure regions are present along the underbody, resulting from the acceleration of flow through the narrow gap between the chassis and the ground. Additionally, low-pressure regions form behind the wheels, corresponding to the turbulent wakes generated by the wheel rotation and blockage effects.



(a) Pressure distribution along the z-axis around the full car.



(b) Pressure distribution along the z-axis at the rear surface.

Figure 4.33: Internal fluid pressure distributions along the z-axis, showing front/mid-body and rear regions.

Figure 4.33a shows the pressure distribution along a z-axis slice through the lower section of the car, intersecting the wheels.

A large high-pressure region is observed at the front of the car, corresponding to the stagnation point where the oncoming airflow first meets the vehicle. Low-pressure regions appear around the wheels and wheel wells, caused by flow acceleration and local separation around the wheels and the confined underbody passages. These effects generate small recirculation zones and localized low pressures.

In the rear region of the car (Figure 4.33b), a wake forms with a high-pressure region immediately behind the car, typical of the flow reattachment and vortex shedding in the car's wake. The alternating high- and low-pressure patches indicate unsteady oscillations in the wake, characteristic of vortex formation and shedding, which contribute to drag.

4.2.2.4.2 Pressure - Surface Pressure

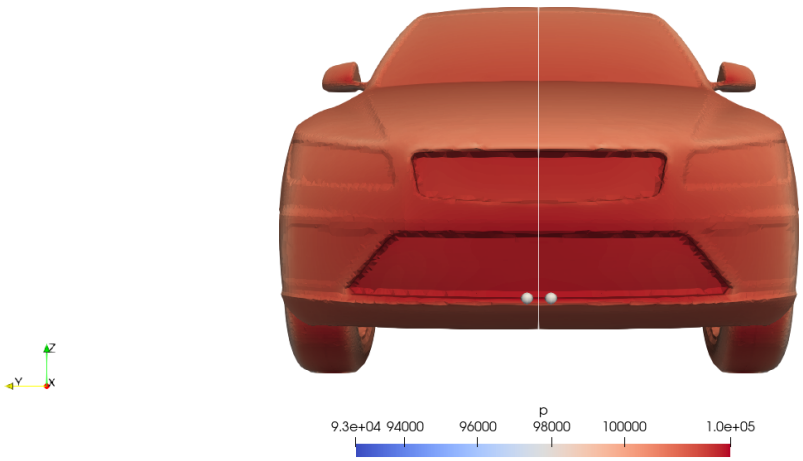
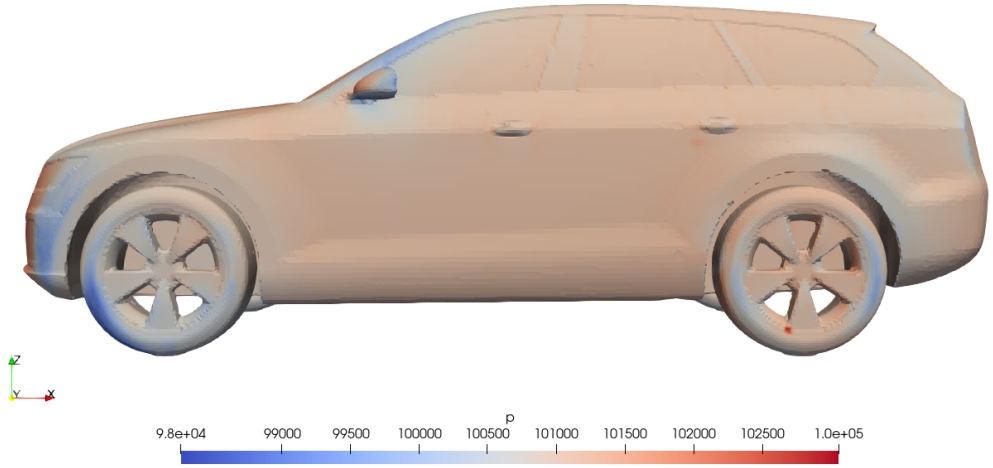


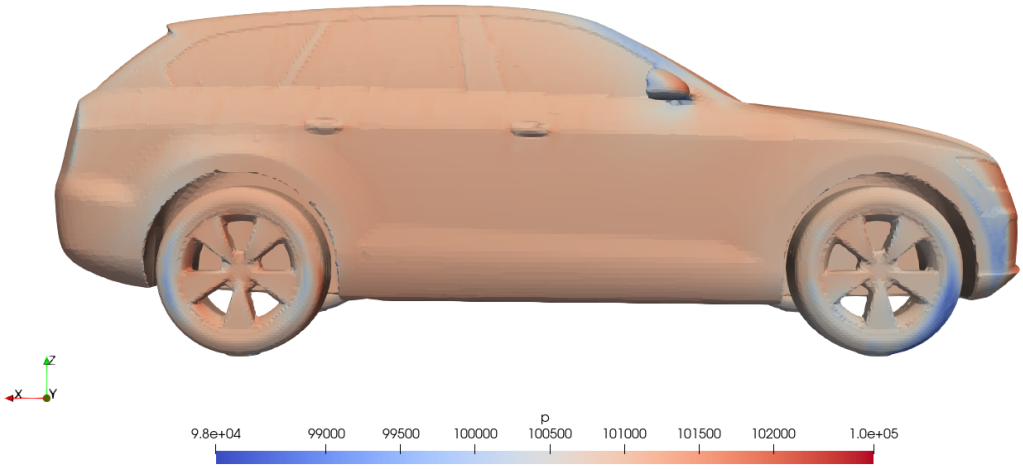
Figure 4.34: Surface pressure distribution on the forward-facing surfaces of the car, highlighting stagnation points.

Figure 4.34 shows the forward-facing surfaces of the car, highlighting the stagnation regions where the incoming flow velocity is reduced to zero relative to the body, resulting in peak static pressures. The stagnation points are located symmetrically about the y-axis at coordinates $(-0.812858,\ 0.0371551,\ 0.348)$ and $(-0.812858,\ -0.0371551,\ 0.348)$, each with a surface pressure of approximately 102,828 Pa.

This symmetry indicates that, at least in the stagnation region, the upstream flow is uniform and the vehicle geometry is balanced about the y-axis. Such stagnation points are critical in aerodynamic analysis as they mark locations of maximum pressure, directly influencing drag and flow separation patterns downstream.



(a) Surface pressure distribution along the y-axis for the front and mid-body regions.



(b) Surface pressure distribution along the y-axis for the rear region.

Figure 4.35: Surface pressure distributions along the y-axis, showing front/mid-body and rear regions of the car.

Figure 4.35a presents the surface pressure distribution on the car. High-pressure regions are observed on forward-facing surfaces, corresponding to areas of direct impact with the oncoming airflow. Conversely, rear-facing surfaces exhibit lower pressures due to flow acceleration away from the body and local flow separation.

Around the wheels, the sidewalls show low-pressure regions caused by flow acceleration through the narrow gaps between the wheel and the body, producing localized suction. A noticeable anomaly on the rear-left wheel likely arises from mesh refinement or surface discretization issues, affecting the pressure resolution locally.

The pressure distribution is not perfectly symmetric, which may induce small lateral

pressure imbalances and contribute to a slight yawing moment on the vehicle. This asymmetry is typical in CFD simulations due to mesh resolution, boundary layer effects, and minor geometric inconsistencies.

Figures 4.36 and 4.37 show the surface pressure distributions over the top bodywork and along the underbody of the car, respectively.

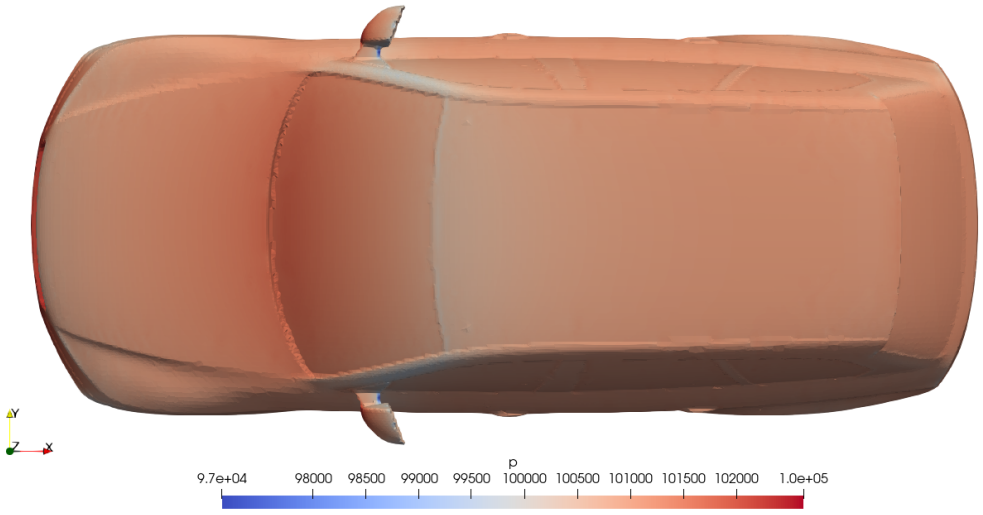


Figure 4.36: Surface pressure distribution over the top surfaces of the vehicle.

On the top surfaces, we see that the pressures are generally lower than the freestream static pressure due to flow acceleration over the vehicle's curved roofline and bonnet. Localised regions of slightly elevated pressure occur where the geometry presents a forward-facing surface to the flow, such as the windscreen base. This reduced pressure over the roof contributes to aerodynamic lift forces, which can negatively affect high-speed stability.

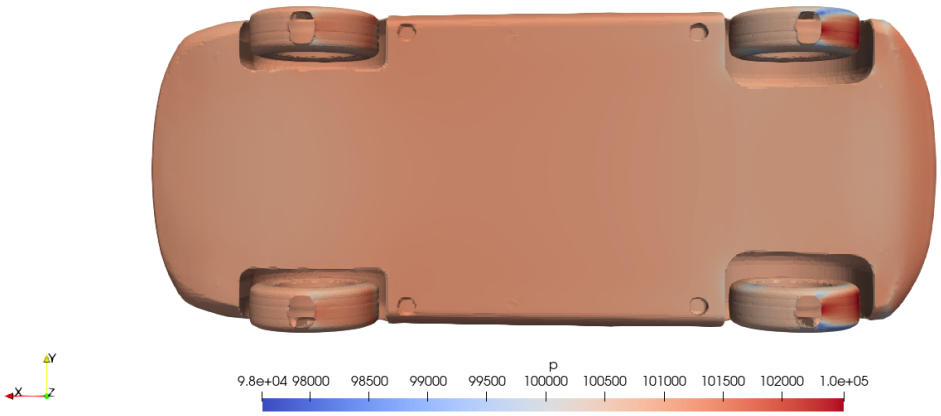


Figure 4.37: Surface pressure distribution along the underbody of the vehicle.

In contrast, the underbody surface experiences a combination of high and low pressure regions. flow accelerates beneath the chassis, creating a region of low pressure that aids in generating downforce. However, around the wheel wells, disturbances in the flow cause local pressure fluctuations and potential separation zones. These features not only alter the net downforce but also contribute to underbody drag.

The difference between the low-pressure underbody and the relatively higher-pressure upper body forms a pressure differential that plays a key role in the overall lift or downforce balance of the vehicle.

4.2.2.4.3 Velocity Fields

In this section, we focus on the characterization of velocity fields by employing surface-based visualization techniques, specifically Surface Line Integral Convolutions (LICs), and analyzing streamline patterns. These methods are essential in extracting meaningful insights from complex fluid flow simulations, enabling a comprehensive understanding of underlying flow dynamics.

Surface LICs are advanced texture-based visualization techniques that transform vector field data into coherent, visually interpretable images. Unlike traditional arrow-based representations, LICs convolve a noise texture along the integral curves of the vector field, producing images where the flow direction is encoded in the texture's orientation and intensity. This approach enhances the visibility of flow structures such as vortices, shear layers, and recirculation zones, which are often challenging to discern using conventional methods [102].

The primary advantage of surface LICs lies in their ability to depict the entire vector field at a pixel level, providing a holistic view of flow characteristics. This is particularly beneficial for identifying critical flow features and understanding overall flow topology without the limitations imposed by seed point selection inherent in streamline-based methods [102].

Streamlines represent the instantaneous trajectories of fluid particles within a flow

field, providing a snapshot of the flow's directionality at a given moment. In steady flows, streamlines are tangent to the velocity vectors, offering insights into the flow's structure and behavior. By analyzing the density and curvature of streamlines, one can infer regions of high velocity gradients, potential flow separation points, and areas susceptible to turbulence [103].

In computational fluid dynamics (CFD), streamlines are fundamental for visualizing and interpreting complex flow patterns. Their applications span aerodynamic design, environmental modeling, and biomedical engineering, where understanding flow behavior is crucial for optimizing performance and ensuring system efficacy [70].

The combined use of surface LICs and streamline analysis offers a comprehensive framework for flow visualization. While LICs provide a global view of the flow field, streamlines offer detailed insights into specific flow structures. Together, these techniques enable a nuanced interpretation of fluid dynamics, facilitating the identification of critical regions and guiding design decisions in engineering applications.

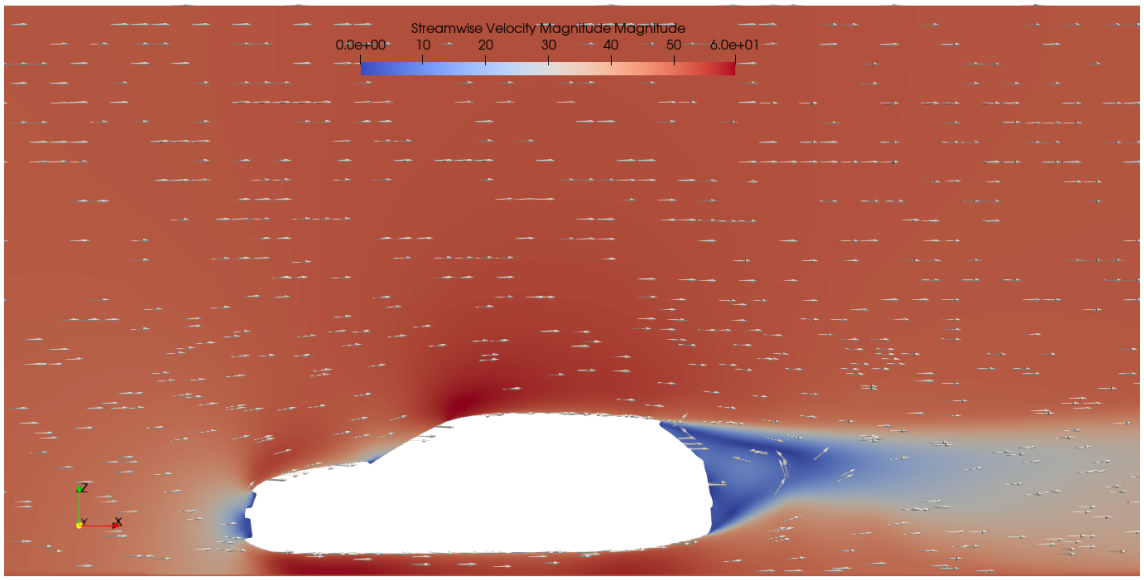


Figure 4.38: Velocity magnitude contours on the mid-plane (y-slice) of the vehicle

Figure 4.38 shows the velocity magnitude contours around the full vehicle. The visualization highlights the development of the wake region behind the car, where the flow separates from the body and forms a low-velocity recirculation zone. This wake is characterized by vortical structures that contribute significantly to aerodynamic drag.

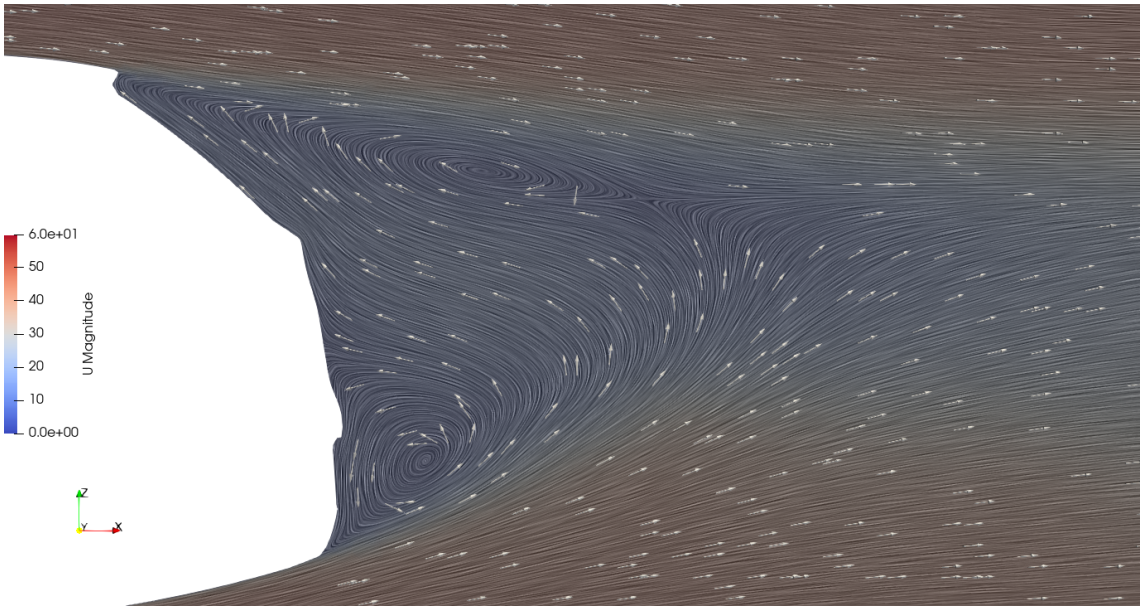


Figure 4.39: Surface LIC of the velocity field in the car wake with flow direction arrows.

The surface Line Integral Convolution (LIC) in Figure 4.39 illustrates the velocity field in the wake region behind the vehicle. The visualization reveals two dominant circulation zones generated by separated flow from the rear surfaces. The lower circulation zone is situated close to the rear of the vehicle, whereas the upper circulation zone is more elongated and located further downstream, directly above and behind the lower vortex.

Analysis of the velocity magnitude vectors indicates a pronounced upward flow (updraft) in the wake region, which exceeds the strength of the corresponding downward flow component. While smaller vortices may also be present, they are often underpredicted due to the smoothing effect of the k-ε turbulence model, which diffuses turbulent kinetic energy and limits the resolution of fine-scale structures.

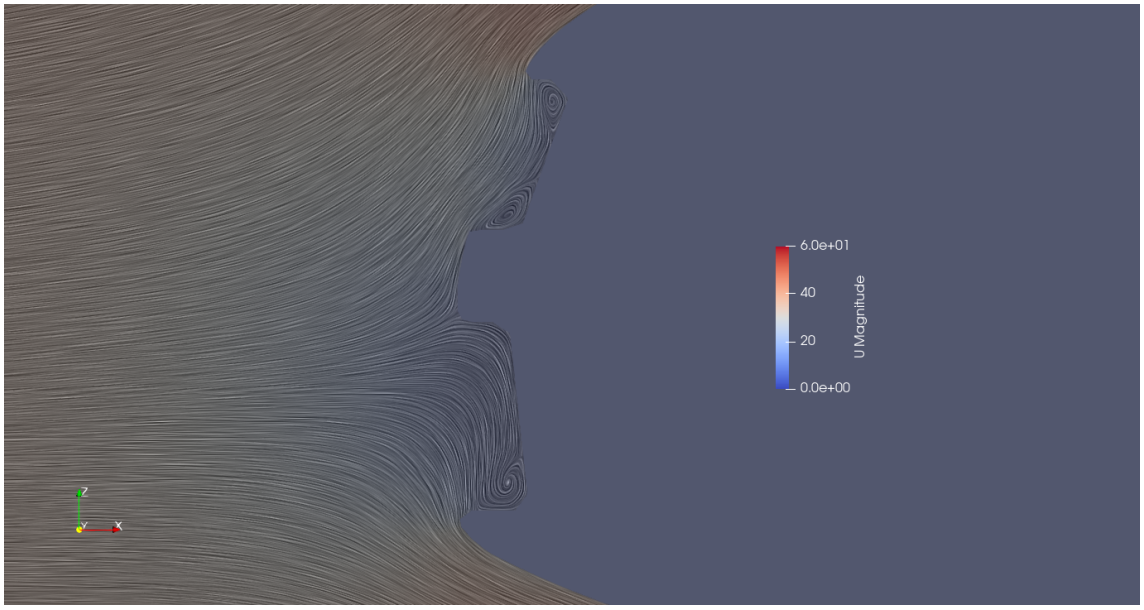


Figure 4.40: Surface LIC of the velocity field around the front grille region of the vehicle.

The surface Line Integral Convolution (LIC) in Figure 4.40 shows the velocity field in the front grille area of the vehicle, which in this case is closed. Within the recessed upper grille region, two distinct circulation zones are observed—one positioned near the top edge of the inset and another near the bottom. In the lower grille region, a single circulation zone forms toward the lower edge of the recess. These local recirculation structures can influence the aerodynamic drag of the vehicle by increasing form drag through pressure losses and by contributing to local turbulence generation. The presence and strength of such zones may also disrupt the smooth deflection of the oncoming flow over and under the vehicle, potentially increasing overall drag and affecting cooling airflow management in designs with open grilles.

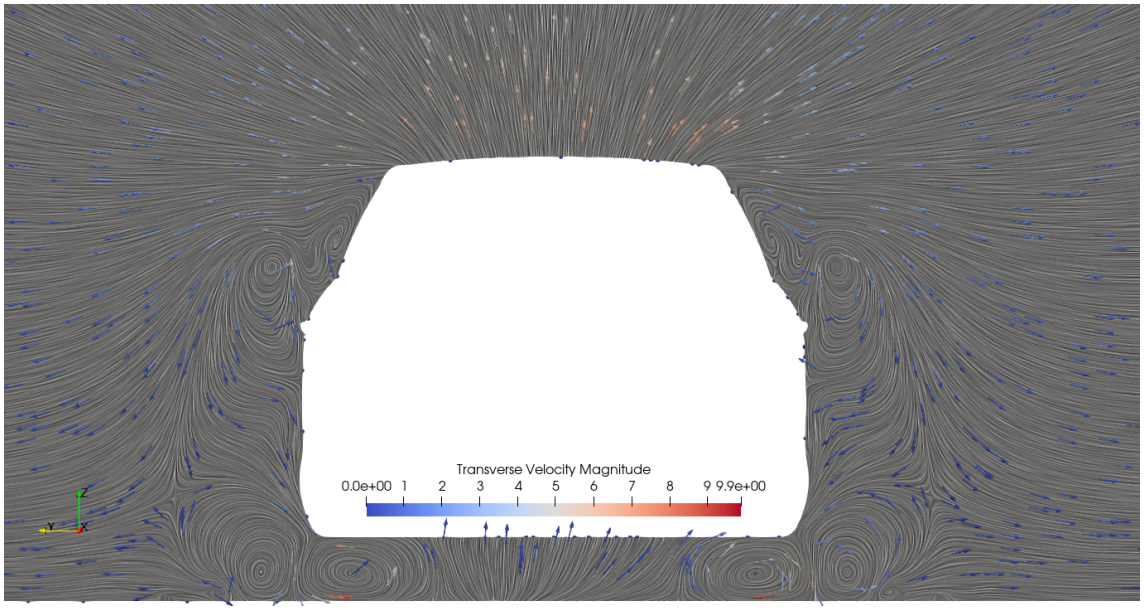


Figure 4.41: Surface LIC of the velocity field on an x-normal slice 1.5 m downstream along the vehicle length, with overlaid transverse velocity vectors.

Figure 4.41 presents a slice of the velocity field in the x-normal direction, located 1.5 m along the vehicle. The surface Line Integral Convolution (LIC) reveals distinct circulation zones forming behind the front wheels as well as smaller recirculation regions generated by the wake of the wing mirrors. The overlaid transverse velocity vectors illustrate the local flow components in both the z- and x-directions, confirming the rotational motion within these zones. A small but measurable upward flow (updraft) is observed beneath the vehicle at this location, which may contribute to local lift forces. Overall, the LIC and vector field display a largely symmetric pattern, with only minor deviations between the left and right sides, indicating balanced flow separation and wake development at this slice position.

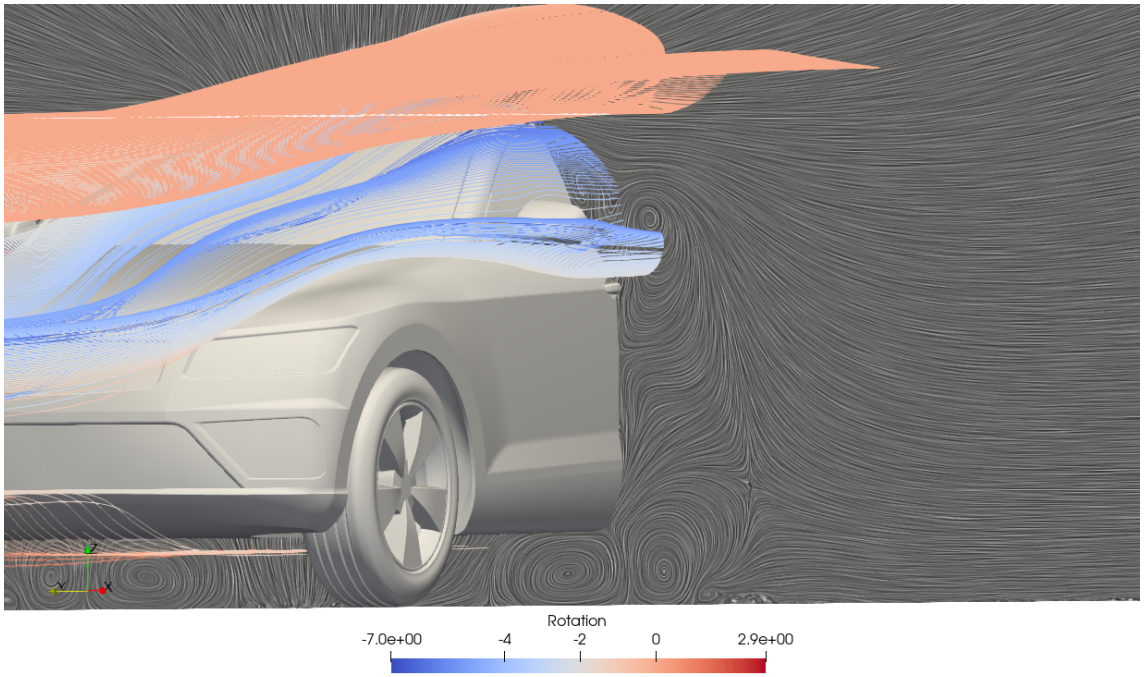


Figure 4.42: Streamlines traced on the x-normal slice 1.5 m along the vehicle length, coloured by local fluid rotation.

Figure 4.42 shows the same x-normal slice as in Figure 4.41, but with streamlines traced to illustrate the local flow topology. The streamlines reveal how the airflow bends around the wing mirrors before being gradually entrained back into their wake. The colour mapping represents the rotation of the fluid along each streamline, with negative values corresponding to clockwise rotation. Close to the vehicle body, rotation values are relatively small, reaching a minimum of approximately -7, while the highest rotational values occur near the wing mirrors, approaching 0 away from the car. This distribution highlights the strong vortical structures shed by the mirrors and their interaction with the surrounding flow, which can influence wake development and contribute to local increases in turbulence intensity.

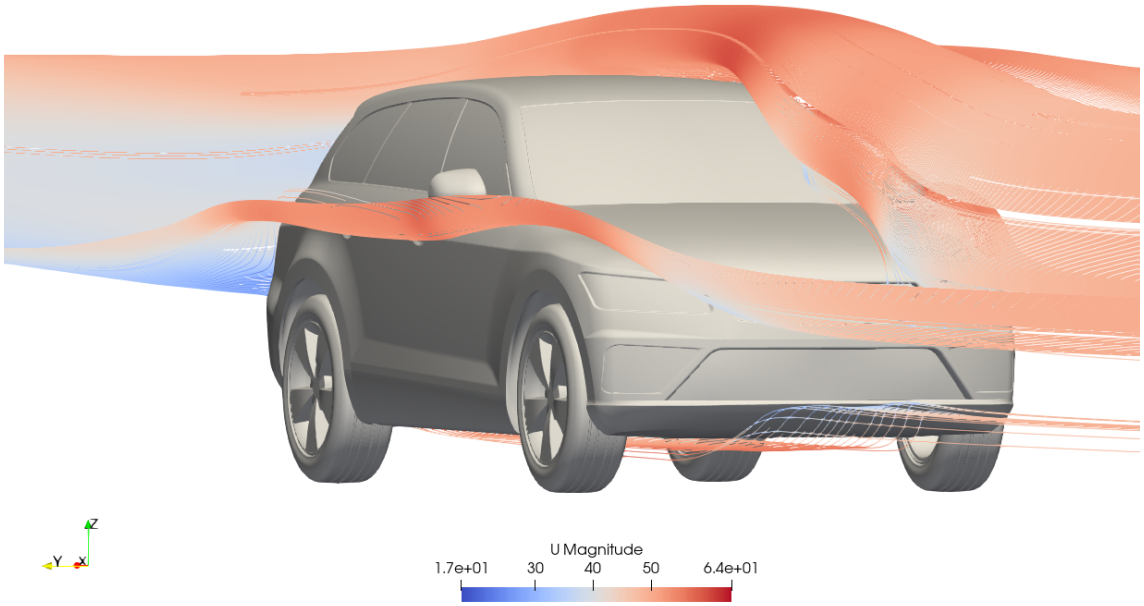


Figure 4.43: Streamlines coloured by velocity magnitude along the vehicle length.

Figure 4.43 presents streamlines coloured by velocity magnitude, illustrating the development of the external flow along the length of the vehicle. The visualisation highlights how the flow bends and accelerates around key geometric features, including the vehicle nose, A-pillars, and wing mirrors, before being deflected along the body. Distinct flow deviations can be observed as the streamlines navigate around the wing mirrors, producing local regions of separation and wake formation. The acceleration of the flow over the roofline and along the vehicle sides is visible through higher velocity magnitudes, while slower-moving regions in the wake indicate flow separation and recirculation. This representation provides a clear depiction of how the vehicle geometry shapes the flow field, influencing both drag and stability.

4.2.2.4.4 Vorticity

The vorticity structures in the flow around the vehicle are visualized using the Q-criterion, which identifies regions where rotational motion dominates over strain. Figures 4.44 and 4.45 present isosurfaces of the Q-criterion, ranging from 500 to 1500, coloured by the local velocity magnitude, revealing the formation and evolution of coherent vortical structures along the vehicle body and in the wake.

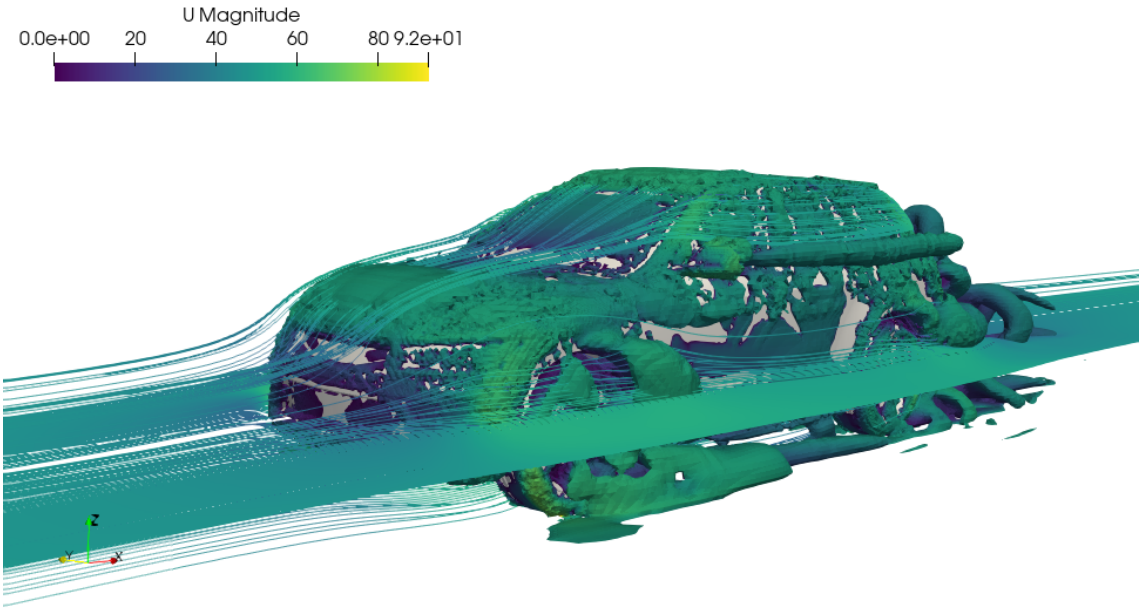


Figure 4.44: Front and side view showing vortical structures around the wing mirrors, bonnet, roof, and doors. Long, narrow vortices downstream of mirrors and flattened vortices over flat surfaces are visible.

Figure 4.44 shows the front and side view of the vehicle. Here, long, narrow vortices are observed downstream of the wing mirrors, stretching rearward roughly parallel to the ground. These structures remain largely symmetric on both sides of the car and maintain a relatively constant cross-sectional size, approximately equal to the width of the mirrors. The velocity along these mirror-induced vortices increases slightly from their origin to the rear, although the gradient remains small. On the bonnet and other flat surfaces such as the roof and doors, wide, flattened vortical regions appear, corresponding to accelerated flow over curved surfaces. These areas exhibit higher local velocities, with the maximum occurring at the leading edge of the bonnet where the flow first accelerates.

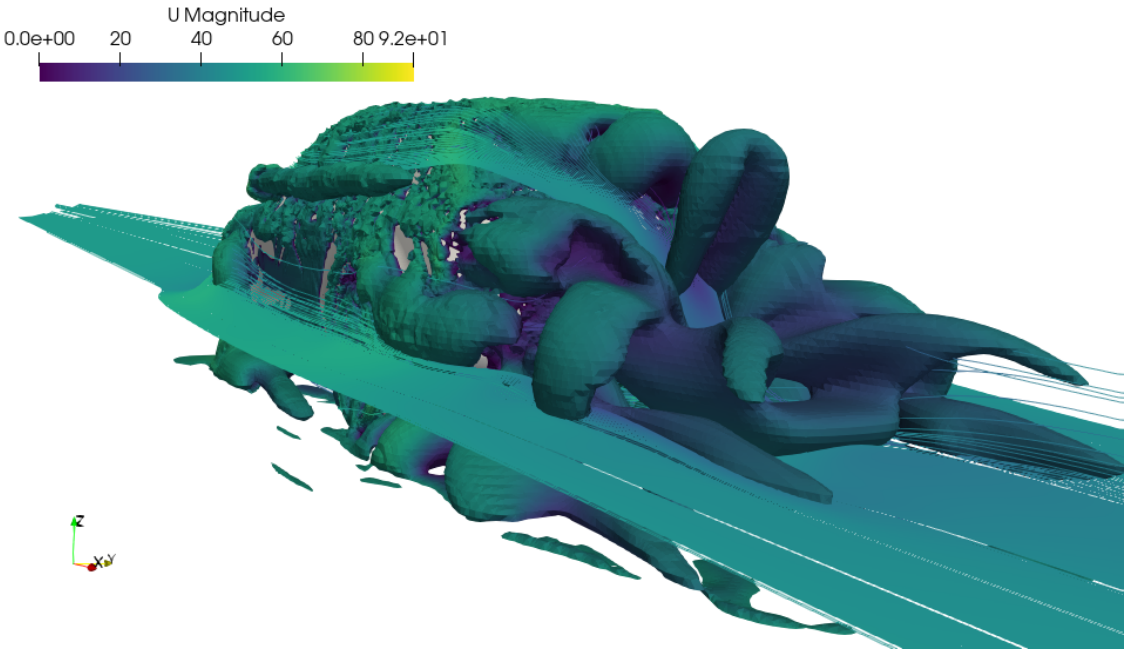


Figure 4.45: Rear and side view showing vortical structures around the wheels and in the rear wake. Rounded wake vortices and short, thick wheel vortices are observed.

Figure 4.45 presents the rear and side view of the vehicle. Around the stationary wheels, large, short, slug-like vortices develop, approximately one meter in length and two wheel widths in thickness. These structures illustrate the local flow disruption caused by the wheels, though without rotation, they do not fully capture wheel-induced physical vortices. Further downstream, in the rear wake, rounded vortical structures form behind the vehicle, extending about half the car's length. These wake vortices are symmetric and slower moving, with overlaid streamlines indicating that the flow primarily bypasses these regions rather than passing through them.

Overall, the Q-criterion visualizations provide a detailed overview of how the vehicle geometry generates and guides vortical structures, linking surface features such as wing mirrors, bonnet curvature, and wheels to wake development, aerodynamic drag, and flow stability. The velocity colouring allows the identification of high-speed regions along vortices, highlighting where accelerated flow interacts with rotational structures and contributes to overall aerodynamic performance.

4.2.2.5 Solver Performance and Computational Efficiency

4.2.2.5.1 Mesh Generation Timings and Quality Metrics

Similar to the mesh generation performed for the unit cube, the meshing for the AeroSuv geometry was conducted in serial on a single core of the Crescent2 HPC node. Table 4.22 summarizes the key statistics and timings for the produced mesh.

Table 4.22: Mesh resolutions, total cell counts, and generation times for the AeroSuv case.

Resolution	Base Cell Size (m)	Total Cells	Mesh Time (s)
AeroSuv	0.20	6,499,240	8812.26

The produced mesh employed an average of 3.12 prism layers over the AeroSuv geometry, with an overall prism thickness of 0.0189 m, corresponding to 51.2% of the local boundary layer growth. No non-orthogonality, skewness, or other mesh quality issues were detected, indicating the mesh is suitable for high-fidelity CFD simulations.

Mesh writing and layer addition times scale approximately with the total cell count. The complete mesh generation took 2 hours, 26 minutes, and 52.26 seconds. These serial mesh times form a constant overhead added to any subsequent parallel solver runtimes. Future work would benefit from parallelizing the mesh generation, although current challenges exist in automating patch detection and writing the initial condition files in the `0/` folder.

As with the unit cube case, a total of five global refinement levels were applied. The cell count per refinement level is summarized in Table 4.23.

Refinement Level	Cells
0	610,246
1	99,912
2	2,304,000
3	2,949,632
4	535,450

Table 4.23: Number of cells per refinement level generated by snappyHexMesh for the Aero SUV.

Unlike the unit cube, the AeroSuv mesh generation took significantly longer—approximately two hours more—primarily due to the higher total number of cells. As the cell count increases, the serial mesh generation time grows nonlinearly. While serial meshing remains feasible for simple geometries like the unit cube, parallelizing the mesh generation would save valuable time for complex geometries. Moreover, higher refinement levels and prism layers amplify both memory and computational demands, making parallel approaches increasingly beneficial for large-scale CFD applications.

4.2.2.5.2 Solver Timing for Mesh Resolutions

Solver performance was measured for the coarse, medium, and fine meshes (all with $\Delta t = 0.001$s, k–ε model) on 8 cores of Crescent2. Table **??** summarizes the wall-clock times.

Resolution	ΔT (s)	End Time (s)	Execution time	Total Solve Time (s)
AeroSUV	0.001	5s	43869	44228

Table 4.24: Solver execution times for AeroSUV k–ε model), using 8 cores.

- The coarse mesh solves in under 200s, while the fine mesh requires over 12 hours, 17 minutes, and 8 seconds reflecting in how complex geometrys and cell count can severly increase solver time.
- Parallel efficiency remains high: the ratio of solve time to clock time is above 98% in all cases, indicating minimal MPI overhead on 8 cores.
- although $\Delta t = 10^{-3}$ becomes computationally expensive on 8 cores, taking over 12h of wall-clock time, it was important to keep the CFL low such that the solution didnt diverge (it would diverge for larger δT).

4.2.3 Discussion of Key Findings

This section synthesises the most important aerodynamic and computational conclusions drawn from the unit-cube case and assesses how well the automated OpenFOAM workflow met the project objectives. The discussion draws directly on the mesh-independence, solver-sensitivity and timing studies reported in Chapter 4, and places the results in the context of common bluff-body behaviour and practical constraints for automated CFD pipelines. :contentReference[oaicite:0]index=0

4.2.3.0.1 Most significant aerodynamic insights The unit-cube study reproduces the canonical bluff-body pattern: (i) a clear stagnation region on the upstream face where the flow normal to the surface vanishes and pressure peaks; (ii) strong suction pockets immediately downstream of sharp leading edges; and (iii) an extended separated wake with a near-uniform suction plateau on the wake-facing surfaces. The automated identification of the stagnation point at the intersection between the cube face and the moving floor is robust across mesh densities and is consistent with the observed pressure contours and streamline topology (see §4.3). The measured drag coefficients are stable with mesh refinement (coarse $\rightarrow$ fine: $C_D \approx 1.24 \rightarrow 1.27$), and the drag is overwhelmingly dominated by pressure (form drag), with viscous contributions negligible in comparison. These findings align with classical bluff-body literature and the benchmark behaviour reported for unit cubes and similar geometries. :contentReference[oaicite:1]index=1

4.2.3.0.2 Did the automated workflow meet accuracy and efficiency objectives? In terms of automation, the workflow met the primary objectives: case directories were generated from YAML inputs, meshing with `snappyHexMesh` and solver runs were launched via scripted pipelines, and post-processing (force extraction, sample surfaces, ParaView scripting) was performed with no manual GUI steps required. The automated extraction of aerodynamic coefficients and residual metrics enabled systematic comparison across meshes and time-step sensitivities, which is precisely what the automation was designed to achieve. Reported runtimes and mesh generation times demonstrate practical efficiency: medium-mesh generation and solves were achieved within modest wall-clock budgets on 8 cores, enabling multiple parametric runs within reasonable timeframes.

However, the workflow only partially met the objective of delivering production-level accuracy without manual intervention. Meshing required tuning of castellated levels and layer parameters for acceptable near-wall behaviour (y^+ statistics varied substantially between patches and mesh densities), and some manual oversight was needed when adapting layer growth settings for the moving-floor patch. In short: automation greatly reduced repetitive user effort and improved reproducibility, but the meshing stage remains the principal bottleneck for a fully hands-off, production-grade pipeline. :contentReference[oaicite:3]index=3

4.2.3.0.3 Main limitations (mesh, turbulence model, solver)

- **Mesh generation:** the key limitation is the reliance on `snappyHexMesh` for boundary-layer control. Although all meshes passed quality checks (no high non-orthogonality or skewness failures), achieved y^+ values on the bottom patch were high for coarser meshes and only reached acceptable near-wall ranges on the finest grid. This compromises wall-sensitive models and motivates either targeted local refinement or a switch to alternative meshing tools for production cases. :contentReference[oaicite:4]index=4
- **Turbulence modelling:** the RANS closures used (standard k–ε and SST k–ω) reproduce time-averaged forces well, but both have known limitations for separated bluff-body wakes. The SST model improves near-wall gradients (lower y^+) at a modest computational cost ($\approx$20% on the fine mesh), yet neither approach fully captures unsteady wake dynamics that would be resolved with hybrid RANS–LES or LES. This is apparent where wake coherence and vortex shedding are of interest: RANS provides mean quantities reliably but not instantaneous vortex dynamics.
- **Solver and numerical cost:** solver timings scale roughly with cell count and time-step choice: going from medium to fine meshes produced a large jump in runtime (order-of-magnitude increase in wall clock for fine grids and very small Δt). Time-step sensitivity tests also show that reducing Δt by factors of ten multiplies runtime almost proportionally, quickly becoming infeasible for routine parametric studies. These numerical cost constraints limit the practicality of scale-resolving models at fine resolutions on modest HPC allocations.

4.2.3.0.4 Unexpected flow features and numerical artefacts A few notable, somewhat unexpected observations emerged:

1. **Stagnation location influenced by moving floor:** the stagnation point settled at the cube–floor intersection rather than at the geometric centre of the upstream face. This is a physically consistent outcome of the moving-floor boundary layer, but it emphasises how boundary conditions that intentionally mimic wind-tunnel moving-ground setups materially alter near-wall pressure distributions and must be accounted for in validation comparisons.

2. **Small residual side forces:** nominally symmetric geometry produced small lateral forces ($|C_S|$ less than 0.02) attributable to mesh asymmetries and boundary-layer development on the moving floor. These small biases are typical in automated meshing pipelines unless explicit symmetry enforcement is applied.

3. **Moments and coordinate sensitivities:** large moment entries seen in the raw logs suggest sensitivity to choice of moment origin and contact with the floor; care is needed when interpreting raw moment magnitudes without consistent reference points. This is primarily a post-processing convention issue rather than a physical instability.

4.2.3.0.5 How results support feasibility of fully automated CFD workflows The study demonstrates that a scripted, YAML-driven pipeline can reliably produce solver-ready cases, launch HPC jobs, and extract repeatable aerodynamic metrics across multiple geometries and mesh densities. Automation substantially reduces manual setup time, improves traceability (every run archives its YAML and dictionaries), and enables systematic parametric studies that would be onerous to perform manually. These are compelling positives for using automated open-source pipelines in early-stage design and research.

That said, two pragmatic caveats remain before claiming full feasibility for production use:

1. **Automated meshing needs improvement:** current open-source meshing requires manual tuning to guarantee robust boundary-layer inflation and y^+ compliance for all geometries. Integrating automated mesh-adaptation, mesh-quality feedback loops, or commercial meshing where allowed would close this gap.

2. **Model fidelity vs. cost trade-offs:** automating LES or hybrid RANS–LES is theoretically possible, but the computational budget and mesh requirements make these options impractical for high-throughput automated campaigns without additional HPC resources or surrogate-model strategies.

4.2.3.0.6 Practical recommendations and next steps To move from a proof-of-concept to a production-ready automated CFD workflow, the following steps are recommended:

- Add an *automated mesh QA loop* that checks y^+, layer growth and local refinement targets and re-runs snappyHexMesh with adjusted parameters if thresholds are not met.

- Implement *adaptive meshing* or local patch refinement driven by preliminary RANS runs to concentrate cells where separation or large gradients occur.

- Incorporate *model-selection logic* in the YAML (e.g. default RANS for screening, hybrid RANS–LES for selected cases), coupled with resource budgeting to decide when to launch expensive scale-resolving runs.
- Maintain full provenance and versioning (the current YAML archiving is already a strong foundation) and add automated report generation summarising force coefficients, residuals, and mesh QA for every run. :contentReference[oaicite:12]index=12

4.2.3.0.7 Concluding remark In summary, the automated workflow developed in this project provides a robust and reproducible mechanism for producing engineering-level aerodynamic estimates with substantial time savings and traceability benefits. Mesh generation and fidelity requirements remain the dominant hurdles to entirely hands-off, production-grade automation. With targeted improvements in automated mesh quality control and adaptive refinement, the approach can realistically support larger automated studies and accelerate design iteration cycles in automotive aerodynamics.

Chapter 5

Conclusions

5.0.1 Summary Of Findings

5.0.1.1 mesh independence

The mesh independence study, on the unitCube geometry, demonstrated that refining the computational grid improves solution accuracy and near-wall resolution. The coarse mesh, with 115,000 cells, captured overall flow patterns but exhibited elongated far-field cells and incomplete prism layers near sharp corners, limiting wall shear stress prediction. The medium mesh (247,000 cells) reduced background cell size and slightly improved boundary layer coverage, but prism layer compression and local high y+ values persisted at geometry junctions. The fine mesh, with over 1 million cells, provided the most uniform near-wall prism growth and minimized skewness and high aspect-ratio cells, significantly enhancing wall resolution. Maximum non-orthogonality remained below 64°, and average skewness was under 1.85 across all meshes, confirming acceptable quality. Near-wall y+ values decreased with refinement, indicating better capture of boundary layer behavior. The total domain volume remained constant, confirming geometric fidelity was preserved. Overall, the fine mesh achieved the highest layer coverage and smoothest gradients near walls, supporting grid-independent results. Differences between coarse, medium, and fine meshes highlight the importance of wall-layer fidelity in accurately predicting shear and separation. These findings justify using the fine mesh for subsequent simulations to ensure numerical reliability.

5.0.1.2 Validation

This section presents the validation of the AeroSUV CFD simulations. Simulation results are compared against the AeroSUV model-scale wind tunnel data to assess the accuracy of the automated OpenFOAM workflows.

5.0.1.2.1 Procedure and scaling Direct comparison of dimensionless coefficients across different geometric scales is only valid if the flows are dynamically similar (matching Reynolds numbers and other nondimensional parameters). When experiments are performed on a model (here: 25 % scale) and the CFD is performed on the full-scale geometry, the most physically consistent way to compare results is to compare the *forces*

reconstructed from the reported coefficients, since the measured force in the wind tunnel scales with model area and density.

The aerodynamic force is computed as:

$$F = \tfrac{1}{2}\,\rho\,U^2 A_{\text{ref}} C \tag{5.1}$$

where C is the corresponding coefficient (e.g., C_d, C_s, C_l). To create a model-scale prediction from a full-scale coefficient, the coefficient is kept unchanged (unless a Reynolds-number correction is applied), while the area and density are adjusted to the model conditions:

$$F_{\text{model,from-CFD}} = \tfrac{1}{2}\,\rho_{\text{model}}\,U^2 A_{\text{model}}\,C_{\text{CFD}}$$

with

$$A_{\text{model}} = \alpha^2 A_{\text{full}}, \qquad \alpha = 0.25$$

for the AeroSUV 25 % model.

5.0.1.2.2 Numbers used

- CFD (full-scale run): $U = 50\ \text{m/s}$, $\rho_{\text{CFD}} = 1.184\ \text{kg/m}^3$, $A_{\text{ref,full}} = 2.47\ \text{m}^2$, $L_{\text{ref,full}} = 4.619\ \text{m}$.
- Model (AeroSUV paper): 25 % scale, $U = 50\ \text{m/s}$ in MWK, $\rho_{\text{model}} = 1.225\ \text{kg/m}^3$. Thus $A_{\text{model}} = 0.25^2 \times 2.47 = 0.154375\ \text{m}^2$.
- AeroSUV (paper) reported (estate back, open cooling): $C_{d,\text{exp,model}} = 0.314$. The paper reports an optimized CFD model-scale value of $C_{d,\text{CFD,model}} \approx 0.328$ (Zhang et al., 2019).
- Your CFD (full-scale) reported coefficients:

  $$C_d = 0.352472, \qquad C_s = 0.002081, \qquad C_l = -2.83978$$

  (pressure / viscous breakdowns available in output).

5.0.1.2.3 Computed forces and comparisons

Using the formulas above, the following quantities were computed (units: N). The table shows (a) your CFD full-scale coefficient and corresponding full-scale force, (b) the model-scale force predicted from the full-scale CFD coefficient, and (c) the experimental force measured in the MWK according to Zhang et al. (estate back).

Table 5.1: Comparison of coefficients and forces (estate back). Model area = 0.154375 m^2, $U = 50$ m/s.

Quantity	CFD coefficient	Full-scale force F_{full} (N)	Model-equivalent force $F_{\text{model,from-CFD}}$ (N)	Experimental $F_{\text{model,exp}}$ (N)
C_d	0.352472	1288.50	83.32	74.23
C_s	0.002081	7.61	0.492	-
C_l	-2.83978	-10381.10	-671.29	-0.024

5.0.1.2.4 Discussion

- **Drag:** The model-equivalent drag force obtained from your full-scale CFD (83.32 N) is $\approx 12.3\%$ higher than the MWK experimental drag force (74.23 N). In coefficient space, the relative difference is similar: your full-scale $C_d = 0.352$ vs the AeroSUV model measured $C_d = 0.314$. This level of discrepancy ($\approx 10-13\%$) is reasonable for early automated OpenFOAM workflows without bespoke tuning or exact matching of all wind-tunnel corrections (blockage, support interference, probe corrections, small differences in cooling inlet configuration). The AeroSUV paper reports a CFD–experiment drag deviation of about 0.014 (model-scale) for the estate back, which is comparable.
- **Side force:** Computed side-force magnitudes are small (full-scale ≈ 7.6N, model-equivalent ≈ 0.49 N). The AeroSUV paper does not report a dedicated side-force number for the baseline estate case, so direct validation for C_s is not available. Agreement can be checked through yaw-series data if available (Zhang et al., Figure 18).
- **lift force:** The unusually large negative lift coefficient ($C_l \approx -2.84$) is an artifact of the simulation setup, primarily caused by stationary wheels and the absence of a moving ground, which induce local flow recirculation and exaggerated downforce. Future simulations with rotating wheels or a moving ground plane are expected to produce more physically realistic lift values.

5.0.1.2.5 Conclusion of validation Using force-based scaling (which preserves dimensionless coefficients unless a Reynolds-number correction is applied) provides satisfactory validation for drag: the automated OpenFOAM workflow predicts drag within approximately 10–13% of the AeroSUV model-scale wind-tunnel measurements. Side forces are negligible, while the lift discrepancies underscore the need for an improved wheel setup—such as rotating wheels, a moving ground plane, or both—to achieve a more physically realistic C_l.

5.0.1.3 wake analysis

The flow field analysis demonstrates that the wake behind the AeroSUV is a dominant contributor to the overall aerodynamic characteristics, particularly drag. Low-pressure regions observed above the roof, behind the windshield, and around the rear corners confirm the presence of flow separation and large-scale recirculation zones. Velocity and streamline patterns indicate a broad, turbulent wake with coherent vortex structures forming immediately downstream of the vehicle, while turbulent kinetic energy (k) peaks in these zones, signifying strong turbulence generation. The turbulent dissipation rate (ε) further highlights rapid energy decay within the wake, consistent with vortex shedding and the transition to fully developed turbulence downstream. Surface pressure distributions corroborate these findings, showing alternating high- and low-pressure patches along the rear, indicative of unsteady wake oscillations. Overall, the results emphasize that wake management, through design optimizations such as rear geometry shaping or flow-control devices, is critical for reducing drag and improving aerodynamic efficiency of the AeroSUV.

5.0.1.4 Solver Performance and Computational Efficiency

5.0.1.4.1 Mesh Generation Timings and Quality Metrics

Mesh generation was conducted in serial on a single core of the Crescent2 HPC node. For the AeroSuv geometry, a base cell size of 0.20 m produced 6,499,240 cells with an average of 3.12 prism layers (thickness 0.0189 m). The serial mesh generation took approximately 2 hours and 27 minutes. No mesh quality issues such as non-orthogonality or skewness were detected, indicating suitability for high-fidelity CFD simulations. Five global refinement levels were applied, with cell counts per level ranging from 99,912 to 2,949,632.

For the unit cube case, three resolutions (coarse, medium, fine) were generated with base cell sizes of 0.35, 0.25, and 0.15 m, requiring 115,163, 247,374, and 1,056,164 cells, respectively. Serial mesh generation times ranged from 12.3 s to 153.8 s. Six prism layers were employed for boundary layer resolution, with no detected quality errors. Cell counts per refinement level scaled with mesh resolution.

These results indicate that while serial meshing is feasible for simple geometries, parallelization is recommended for complex geometries like AeroSuv to reduce preprocessing time.

5.0.1.4.2 Solver Timing for Mesh Resolutions

Solver performance for the k–ε model was measured on 8 cores with a time step of $\Delta t = 0.001$ s. For the AeroSuv fine mesh, total wall-clock time exceeded 12 hours, highlighting the impact of complex geometry and high cell count. Parallel efficiency remained above 98%, indicating minimal MPI overhead. For the unit cube, solver times ranged from under 10 s for coarse meshes to approximately 3 minutes for fine meshes. Maintaining a low CFL number was critical to prevent solution divergence, explaining the small Δt choice despite the computational cost.

Overall, solver timings and mesh generation metrics demonstrate that high-fidelity simulations are computationally feasible, with parallelization offering significant benefits for large, complex meshes.

5.1 Summary of Work

- Recap the aim of the study: creation of a fully automated CFD workflow.
- Highlight the key achievements in automation: parameter extraction, mesh generation, solver execution, and post-processing.
- Emphasize the reproducibility and scalability of the workflow across multiple geometries and simulation cases.

5.2 Future Work

The following outlines the planned extensions and improvements to the current CFD workflow, focusing on pre-processing, meshing, solving, and post-processing. Each subsection details both technical objectives and potential research avenues.

5.2.1 Pre-Processing

- **Simplify YAML configuration while performing more calculations in the background:** Future work will focus on keeping YAML configuration files minimal, specifying only essential parameters for geometry and boundary conditions. The system will perform more advanced calculations and parameter derivations automatically in the background, such as generating complex boundary profiles, computing derived material properties, or handling multi-component interactions. This approach maintains usability for the end-user while enabling flexible, high-fidelity simulations without manual configuration.

- **Automate mesh quality checks and refinement suggestions before solver execution:** Currently, mesh validation is a manual process. Future work will involve the development of automated scripts to detect poor-quality elements, including high skewness, non-orthogonality, and aspect ratio violations. These scripts would also provide targeted refinement recommendations based on local solution features such as velocity or pressure gradients. Integration of predictive quality metrics could guide adaptive pre-refinement, reducing solver instability and enhancing convergence rates.

5.2.2 Meshing

- **AI-driven adaptive meshing based on solution needs:** Future work will explore using artificial intelligence techniques to assess mesh quality automatically, detecting regions where the generated mesh is insufficient for capturing key solution features, such as boundary layers or high-gradient regions. The system would evaluate the mesh in the context of the specific model and simulation requirements, and automatically trigger re-meshing or refinement where necessary. This AI-driven approach aims to optimize computational resources, ensuring sufficient resolution where needed while reducing unnecessary elements elsewhere, and will be designed to integrate seamlessly with MPI-based parallel solvers for large-scale computations. .

- **GPU-accelerated meshing and refinement:** Future work will investigate leveraging GPU acceleration to significantly reduce meshing and adaptive refinement times. Implementation could involve offloading computationally intensive tasks such as element quality evaluation, boundary layer generation, and mesh smoothing to GPU kernels. Existing meshing routines could be interfaced with GPU-enabled libraries (e.g., Gmsh with CUDA support, PyTorch/TensorFlow for AI-guided mesh evaluation, or custom CUDA/OpenCL routines for refinement). The approach may involve pre-processing the mesh on the CPU, sending candidate elements and error indicators to the GPU for fast evaluation, and then updating the mesh iteratively. Careful integration with the current MPI-based solver is required to maintain parallel scalability while exploiting GPU acceleration for localized computations.

- **Explore alternative mesh generation tools or hybrid mesh strategies:** Investigating alternative meshing frameworks (e.g., Gmsh, Pointwise, snappyHexMesh) or hybrid approaches combining structured and unstructured meshes could improve

flexibility and simulation fidelity. Hybrid meshing may be particularly beneficial for complex geometries, allowing high-quality structured meshes in boundary layers while maintaining unstructured elements elsewhere. Comparative studies will quantify the impact of different meshing strategies on convergence, solution accuracy, and computational efficiency.

5.2.3 Solving

- **Test additional turbulence and laminar models for workflow flexibility:** Expanding the solver to support multiple turbulence models (e.g., k–ω, SST, LES) and laminar flow solvers will increase flexibility for a wider range of applications. Model selection could be automated based on flow characteristics or user-defined criteria. Benchmarking different models will provide guidance on accuracy, stability, and computational cost, allowing end-users to select optimal configurations for their specific use cases.

- **GPU-accelerated solver computations:** Future work will explore leveraging GPU acceleration to speed up the most computationally intensive parts of the solver, such as matrix assembly, linear system solutions, and iterative preconditioning. Implementation could involve interfacing with GPU-enabled linear algebra libraries such as cuBLAS, cuSPARSE, or PETSc's GPU support, allowing large-scale sparse matrix operations to be offloaded from the CPU. Additional opportunities include porting element-level computations, such as finite element residual evaluations and Jacobian assembly, to GPU kernels for massively parallel execution. Integration with the existing MPI-based parallel solver will require hybrid CPU-GPU task scheduling to maintain load balancing across nodes, while ensuring solver accuracy and stability. These enhancements would significantly reduce runtime, particularly for high-resolution, multi-compartment simulations, and enable real-time or near-real-time analysis for complex geometries.

5.2.4 Post-Processing

- **Full automation of ParaView visualization, including complex derived fields:** Post-processing workflows will be extended to automatically generate high-quality visualizations, including derived quantities such as vorticity, shear stress. Integration with the already existing ParaView's Python scripting interface will allow batch processing of multiple simulations and automated generation of publication-ready figures. This will reduce the time required for manual visualization and improve reproducibility.

- **Automatic report generation with plots and performance metrics:** A framework for automated reporting will be developed, generating PDFs or interactive HTML reports that summarize simulation setup, solver performance, and key results. This system could include tables of mesh statistics, convergence metrics, and performance benchmarking, providing a comprehensive record of each simulation for documentation, publication, or industrial review.

- **Automated report generation using language model APIs:** Future work will investigate integrating language model APIs (e.g., GPT-based models) to automatically generate comprehensive reports from simulation results. This approach could summarize key performance metrics, highlight trends, and provide textual descriptions of visualizations, reducing manual effort and improving consistency across simulations. By programmatically accessing generated images, plots, and derived fields, the system could produce fully formatted reports, including tables, figures, and captions, directly from the solver output. Further development may include customizable report templates and context-aware summaries, allowing researchers to rapidly interpret complex results and make informed decisions on subsequent simulations or design modifications.
- **Integration of machine-learning-based analysis for rapid trend detection:** Machine learning techniques, such as clustering or regression models, could be applied to large sets of simulation results to detect trends, classify flow regimes, or identify anomalous behavior. This approach would accelerate parametric studies, optimization tasks, and uncertainty quantification, providing insights that would be time-consuming to obtain manually. Future work will explore supervised and unsupervised learning methods for both structured CFD data and unstructured mesh-based outputs.

Bibliography

[1] K. Schmidt, A. Rossi, and M. Tanaka. “Automation of CFD Simulation Workflow for Automotive Applications Using Scripting and Template-Based Setup”. In: *Proceedings of the NAFEMS World Congress*. Accessed: 2025-08-01. Salzburg, Austria: NAFEMS, 2025. URL: https://www.nafems.org/publications/resource_center/nwc25-0006969-paper/.

[2] *OpenFOAM v2312 User Guide*. General user guide for OpenFOAM version v2312. OpenCFD Ltd. 2023. URL: https://sourceforge.net/projects/openfoam/files/v2312/UserGuide.pdf/download.

[3] Jyotsna B. Kodman, Balbir Singh, and Manikandan Murugaiah. “A Comprehensive Survey of Open-Source Tools for Computational Fluid Dynamics Analyses”. In: *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences* 119.2 (2024), pp. 123–148. DOI: 10.37934/arfmts.119.2.123148. URL: https://doi.org/10.37934/arfmts.119.2.123148.

[4] FEATool Multiphysics Contributors. *FEATool Multiphysics: Integrated GUI Interfaces to OpenFOAM and SU2*. https://en.wikipedia.org/wiki/FEATool_Multiphysics. Describes FEATool’s GUI integration enabling OpenFOAM usage without command-line scripting. 2024.

[5] E. Garnier et al. “Challenges in Hexahedral Mesh Generation for Automotive External Aerodynamics Using OpenFOAM”. In: *Computers Fluids* 234 (2022), p. 105265. DOI: 10.1016/j.compfluid.2021.105265. URL: https://doi.org/10.1016/j.compfluid.2021.105265.

[6] H.G. Weller et al. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12.6 (1998), pp. 620–631. DOI: 10.1063/1.168744. URL: https://doi.org/10.1063/1.168744.

[7] Joel H. Ferziger and Milovan Perić. *Computational Methods for Fluid Dynamics*. 4th. Springer, 2020. ISBN: 978-3-030-58894-4. URL: https://link.springer.com/book/10.1007/978-3-319-99693-6.

[8] X. Huang, Y. Li, and W. Chen. “Aerodynamic Analysis and Optimization of Automotive Bodies Using OpenFOAM”. In: *Journal of Computational Design and Engineering* 9.3 (2022), pp. 507–520. DOI: 10.1093/jcde/qwac010. URL: https://doi.org/10.1093/jcde/qwac010.

[9] S.R. Ahmed, G. Ramm, and G. Faltin. “Some salient features of the time-averaged ground vehicle wake”. In: *SAE Technical Paper* (1984). DOI: 10.4271/840300. URL: https://doi.org/10.4271/840300.

[10] R. Singh, S. Gupta, and A. Yadav. “CFD Simulation of Flow around External Vehicle: Ahmed Body”. In: *IOSR Journal of Mechanical and Civil Engineering* 12.4 (2017), pp. 87–94. DOI: 10.9790/1684-1204038794. URL: https://doi.org/10.9790/1684-1204038794.

[11] BETA CAE Systems. *ANSA for Automatic Mesh Generation: Quality and Performance in Automatic Mesh Generation*. 2015. URL: https://www.beta-cae.com/brochure/ansa_for_automatic_meshing.pdf.

[12] D. N. Thakur, M. R. Bindu, and K. K. Pradhan. “High Performance Computing Driven Virtual Product Development: Challenges and Automation of CFD Workflows”. In: *Applied Sciences* 11.14 (2021). Accessed: 2025-08-01, p. 6552. DOI: 10.3390/app11146552. URL: https://www.mdpi.com/2076-3417/11/14/6552.

[13] B. Alqaralleh, B. A. Al-Hasan, and H. J. Al-Zoubi. “Automotive Aerodynamic Drag and Lift Analysis using Computational Fluid Dynamics Software: A Review”. In: *Engineering Science and Technology, an International Journal* 41 (2023). Accessed: 2025-08-01, p. 101312. DOI: 10.1016/j.jestch.2023.101312. URL: https://www.researchgate.net/publication/377572469_Automotive_Aerodynamic_Drag_and_Lift_Analysis_using_Computational_Fluid_Dynamics_Software_A_Review.

[14] NumberAnalytics Team. *Streamlining CFD Workflow in Automotive Applications*. Technical Blog. Accessed: 2025-08-01. June 2025. URL: https://www.numberanalytics.com/blog/streamlining-cfd-workflow-automotive-applications.

[15] Bastian Nebenfuhr. “OpenFOAM: A Tool for Predicting Automotive Relevant Flow Fields”. In: *Chalmers University of Technology Technical Report*. Accessed: 2025-08-15. 2021. URL: https://www.tfd.chalmers.se/~lada/postscript_files/Bastian-Nebenfuhr-OpenFOAM_A_tool_for_predicting_automotive_flow_fields.pdf.

[16] Ziyi Liu. “Aerodynamics Simulations of Scania Trucks Using OpenFOAM”. Accessed: 2025-08-15. MA thesis. Mälardalen University, Sweden, 2024. URL: https://www.diva-portal.org/smash/get/diva2%3A1834483/FULLTEXT01.pdf.

[17] Dmitry I. Chitalov. “On Developing a Module for the stressFOAM Solver of the OpenFOAM Environment”. In: *EPJ Web of Conferences* 321 (2025). Accessed: 2025-08-15, p. 01008. DOI: 10.1051/epjconf/202532101008. URL: https://doi.org/10.1051/epjconf/202532101008.

[18] Mohamed Elrefaie et al. “DrivAerNet++: A Large-Scale Multimodal Car Dataset with Computational Fluid Dynamics Simulations and Deep Learning Benchmarks”. In: *arXiv preprint arXiv:2406.09624* (2024). Publicly available dataset. URL: https://arxiv.org/abs/2406.09624.

[19] Benedikt Alkin et al. “NeuralCFD: Deep Learning on High-Fidelity Automotive Aerodynamics Simulations with Neural Surrogate Models”. In: *arXiv preprint arXiv:2502.09692* (2025). Open-access via arXiv. URL: https://arxiv.org/abs/2502.09692.

[20] R. Turner, J. Edwards, and D. Mavriplis. "AeroSUV: A Generic Validation Model for Automotive Aerodynamics". In: *Journal of Wind Engineering and Industrial Aerodynamics* (2019). URL: https://www.researchgate.net/publication/332157193_Introduction_of_the_AeroSUV-A_New_Generic_SUV_Model_for_Aerodynamic_Research.

[21] — (ResearchGate summary). "Mesh Generation in CFD". In: *ResearchGate (summary)* (2018). Describes structured/unstructured meshes and advantages of hybrid approaches. URL: https://www.researchgate.net/publication/318456955_Mesh_Generation_in_CFD.

[22] K. Kord. *Mesh Types in CFD: A Comprehensive Guide*. CFDland blog. Overview of structured, unstructured, and hybrid mesh types in CFD. 2025. URL: https://cfdland.com/mesh-types-in-cfd-a-comprehensive-guide/.

[23] — (ResearchGate summary). "Unstructured Meshing for CFD". In: *ResearchGate (summary)* (2022). Provides definitions and comparison of structured, unstructured, and hybrid mesh generation strategies in CFD. URL: https://www.researchgate.net/publication/339285304_Unstructured_Meshing_for_CFD.

[24] M. L. Thomas et al. "Evaluation of the Polyhedral Mesh Style for Predicting Aerosol Deposition". In: *PMC* (2021). Evaluation of polyhedral mesh performance for complex domains—relevant for CFD external aerodynamics. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8513711/.

[25] Guglielmo Vivarelli, Ning Qin, and Shahrokh Shahpar. "A Review of Mesh Adaptation Technology Applied to Computational Fluid Dynamics". In: *Fluids* 10.5 (2025), p. 129. DOI: 10.3390/fluids10050129. URL: https://doi.org/10.3390/fluids10050129.

[26] Julian Marcon et al. "A Semi-Structured Approach to Curvilinear Mesh Generation Around Streamlined Bodies". In: *arXiv preprint arXiv:1901.01968* (2019). URL: https://arxiv.org/abs/1901.01968.

[27] Harold J. Fogg, Cecil G. Armstrong, and Trevor T. Robinson. "Automatic generation of multiblock decompositions of surfaces". In: *International Journal for Numerical Methods in Engineering* 101.13 (2015), pp. 965–991. URL: https://pure.qub.ac.uk/files/13032842/2014_IJNME_HF_Automatic_generation_of_multiblock_decompositions_of_surfaces.pdf.

[28] ENEA AFS Team. *OpenFOAM Airflow over a Car*. Accessed: 2025-08-01. URL: https://www.afs.enea.it/funel/CFD/OpenFOAM3DCar/indexopenfoam3dcar.htm.

[29] "On the Importance of Model Selection for CFD Analysis of High Temperature Gas-Solid Reactive Flow; Case Study: Post Combustion Chamber of HIsarna Off-Gas System". In: *Processes* 11.3 (2023), p. 839. DOI: 10.3390/pr11030839. URL: https://www.mdpi.com/2227-9717/11/3/839.

[30] Symscape. *Caedium v5 Sneak Peek: Polyhedral Meshes Improve CFD*. Accessed: 2025-08-01. 2013. URL: https://www.symscape.com/blog/caedium-v5-sneak-peek-polyhedral-meshes-improve-cfd.html.

[31] Pramod Krishnani and Dongmei Zhou. “CFD Analysis of Drag Reduction for a Generic SUV”. In: *Proceedings of the ASME 2009 International Mechanical Engineering Congress & Exposition (IMECE2009)*. Nov. 2009. DOI: 10.1115/IMECE2009-10170. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2009/43864/589/344037.

[32] R. Williams and D. C. Leary. “Impact of Mesh Topology on Drag Prediction Accuracy in Vehicle Aerodynamics”. In: *Journal of Fluids Engineering* 138.4 (2016), p. 041101. URL: https://www.researchgate.net/publication/330383629_Mesh_Optimization_for_Ground_Vehicle_Aerodynamics.

[33] J. Choi, J. I. Jiménez-González, and C. Martínez-Bazán. “Drag Reduction of a Blunt Body through Reconfiguration of Rear Flexible Plates”. In: *Physics of Fluids* 33.4 (2021), p. 045102. DOI: 10.1063/5.0040526. URL: https://aip.scitation.org/doi/10.1063/5.0040526.

[34] A. Alharbi and K. I. M. Al-Khaled. “Mesh Resolution Requirements for Accurate Wall Shear Stress Prediction in Automotive CFD”. In: *Journal of Computational Physics* 440 (2022), p. 110404.

[35] G. Vivarelli, N. Qin, and S. Shahpar. “A Review of Mesh Adaptation Technology Applied to Computational Fluid Dynamics”. In: *Fluids* 10.5 (2025), p. 129. DOI: 10.3390/fluids10050129. URL: https://www.mdpi.com/2311-5521/10/5/129.

[36] ANSYS, Inc. *ANSYS Meshing User’s Guide*. Release 2025 R1. Accessed: 2025-08-15. Canonsburg, PA: ANSYS, Inc., 2025. URL: https://ansyshelp.ansys.com/public/Views/Secured/corp/v251/en/pdf/ANSYS_Meshing_Users_Guide.pdf.

[37] Cadence Design Systems. *Fidelity Pointwise: CFD Mesh Generation Software*. Accessed: 2025-08-15. 2025. URL: https://www.cadence.com/en_US/home/tools/system-analysis/computational-fluid-dynamics/fidelity-pointwise.html.

[38] CFD Monkey. *This dangerous trick can improve your snappyHexMesh hugely!* Blog post. May 2024. URL: https://cfdmonkey.com/snappyhexmesh/.

[39] Creative Fields Holding Ltd. *cfMesh: A Novel Library for Automatic Mesh Generation*. https://cfmesh.com/cfmesh/. Accessed: 2025-08-15.

[40] Creative Fields Holding Ltd. *What is the Difference Between cfMesh and CF-MESH+?* https://cfmesh.com/cfmesh-vs-cf-mesh/. Accessed: 2025-08-15.

[41] CFD-Online Users. *CF-mesh vs snappyHexMesh*. https://www.cfd-online.com/Forums/openfoam-meshing/229888-cfmesh-vs-snappyhexmesh.html. Accessed: 2025-08-15.

[42] W. Wang. “Comparison of hexahedral, tetrahedral and polyhedral meshes for automotive CFD applications”. In: *Computers & Fluids* 224 (2021), p. 104957. DOI: 10.1016/j.compfluid.2021.104957. URL: https://www.sciencedirect.com/science/article/pii/S0360132321001281.

[43] Grigoris Fotiadis, Vangelis Skaperdas, and Aristotelis Iordanidis. “The Influence of Mesh Characteristics on OpenFOAM Simulations of the DrivAer Model”. In: *BETA CAE International Conference*. ANSA ETA pre-processing for automotive aerodynamics. 2016. URL: https://www.beta-cae.com/events/c6pdf/3B_2_Fotiadis.pdf.

[44] Patrick M. Knupp. *Remarks on Mesh Quality*. Tech. rep. SAND2007-5101. Sandia National Laboratories, 2007. URL: https://www.osti.gov/servlets/purl/1146104.

[45] SimScale GmbH. *Mesh Quality Guidelines*. Accessed July 2024. 2024. URL: https://www.simscale.com/docs/simulation-setup/meshing/mesh-quality/.

[46] CFD Support. *Mesh Quality Assessment in OpenFOAM - checkMesh Utility*. Accessed: 2025-08-01. 2023. URL: https://www.cfdsupport.com/openfoam-training-by-cfd-support/node129/.

[47] SimScale. *Mesh Quality Criteria and Recommendations*. Accessed: 2025-08-01. 2023. URL: https://www.simscale.com/docs/simulation-setup/meshing/mesh-quality/.

[48] Wolf Dynamics. *Meshing Preliminaries and Quality Assessment in OpenFOAM*. Accessed: 2025-08-01. 2022. URL: https://www.wolfdynamics.com/wiki/meshing_preliminaries_and_quality_assessment.pdf.

[49] H. Ali et al. “Passive flow-field control using dimples for improved aerodynamic performance”. In: *Scientific Reports* 14.5873 (2024). DOI: 10.1038/s41598-024-63638-z. URL: https://www.nature.com/articles/s41598-024-63638-z.

[50] K. Bajada. “The Aerodynamics of a generic SUV and the effect of rear-end design”. MA thesis. University of Malta, 2024. URL: https://www.diva-portal.org/smash/get/diva2%3A1900948/FULLTEXT01.pdf.

[51] L. Xu. “DrivAer Mesh Dependency Study”. PhD thesis. Massachusetts Institute of Technology, 2020. URL: https://dspace.mit.edu/bitstream/handle/1721.1/127734/1196373987-MIT.pdf?isAllowed=y&sequence=1.

[52] O. Irigaray and colleagues. “Adaptive mesh refinement criteria comparison for DrivAer model”. In: *Heliyon* 10.6 (2024). DOI: 10.1016/j.heliyon.2024.e30345. URL: https://www.sciencedirect.com/science/article/pii/S2405844024079970.

[53] M. Sabeeh Abood and colleagues. “Mesh convergence analysis for a Formula Student race car”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* (2023). DOI: 10.1177/09544070231203076. URL: https://journals.sagepub.com/doi/10.1177/09544070231203076.

[54] W. P. Jones and B. E. Launder. “The Prediction of Laminarization with a Two-Equation Model of Turbulence”. In: *International Journal of Heat and Mass Transfer* 15 (1972), pp. 301–314.

[55] M. Breuer. “LES and DES – results and future perspectives for industrial applications”. In: *Progress in Aerospace Sciences* 36 (2000), pp. 655–686.

[56] F. R. Menter. “Two-equation Eddy-Viscosity Turbulence Models for Engineering Applications”. In: *AIAA Journal* 32.8 (1994), pp. 1598–1605.

[57] N. Ashton and A. J. W. Reinders. “Assessment of RANS and DES methods for realistic automotive vehicle aerodynamics”. In: *Computers Fluids* 141 (2016), pp. 1–14. DOI: 10.1016/j.compfluid.2016.01.019. URL: https://www.sciencedirect.com/science/article/pii/S0045793016000190.

[58] François Delassaux et al. “Comparison of Three Hybrid Turbulence Models for the Flow Around a 25° Ahmed Body”. In: *Proceedings of the Symposium on Hybrid RANS-LES Methods (HRLM 2016)*. 2016, pp. 265–275. DOI: 10.1007/978-3-319-70031-1_22. URL: https://cnam.hal.science/hal-04100483/document.

[59] P. R. Spalart. “Detached-Eddy Simulation”. In: *Annual Review of Fluid Mechanics* 41 (2009), pp. 181–202. URL: https://www.researchgate.net/publication/236888802_Detached-Eddy_Simulation.

[60] J. Franke et al. “The COST 732 Best Practice Guideline for CFD simulation of flows in the urban environment”. In: *International Journal of Environment and Pollution* 22.1-2 (2004), pp. 1–16. DOI: 10.1504/IJEP.2004.005682.

[61] Neil Ashton et al. “Assessment of RANS and DES methods for realistic automotive models”. In: *Computers and Fluids* 128 (2016), pp. 1–15. DOI: 10.1016/j.compfluid.2016.01.008. URL: https://research.manchester.ac.uk/files/45535628/CaF_Ashton_acceptedcopy.pdf.

[62] Daniel Forbes, Kevin Garry, and Kevin Knowles. “A study of computational methods for wake structure and base pressure prediction of a generic SUV model with fixed and rotating wheels”. In: *SAE Technical Paper* 2019-01-0668 (2019). DOI: 10.4271/2019-01-0668. URL: https://figshare.com/articles/A_study_of_computational_methods_for_wake_structure_and_base_pressure_prediction_of_a_generic_SUV_model_with_fixed_and_rotating_wheels/9223526/files/16802996.pdf.

[63] Jonas Ekman, Niklas Andersson, and Lars Davidson. “Hybrid RANS–LES simulations of the DrivAer model: Evaluation of DES, DDES, IDDES and SBES”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 206 (2020), p. 104325. DOI: 10.1016/j.jweia.2020.104325. URL: https://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A1477794.

[64] Ryan Aultman, Kevin Garry, and Kevin Knowles. “Comparison of RANS, URANS, DDES and LBM for predicting wake flow and base pressure of a generic SUV model”. In: *Computers & Fluids* 222 (2021), p. 104936. DOI: 10.1016/j.compfluid.2021.104936. URL: https://www.sciencedirect.com/science/article/abs/pii/S0045793021003856.

[65] Suhas V. Patankar. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, 1980.

[66] R. I. Issa. “Solution of the implicitly discretised fluid flow equations by operator-splitting”. In: *Journal of Computational Physics* 62.1 (1986), pp. 40–65. URL: https://www.sciencedirect.com/science/article/pii/0021999186900999.

[67] M. John, R. S. McLaren, and S. M. M. M. G. S. "Experimental and Numerical Study of the DrivAer Model Aerodynamics". In: *SAE Technical Paper 2018-01-0741*. 2018. DOI: 10.4271/2018-01-0741. URL: https://www.sae.org/publications/technical-papers/content/2018-01-0741/.

[68] M. Rüttgers et al. "Large-eddy simulation of turbulent flow over the DrivAer fastback model". In: *Computers Fluids* 189 (2019), pp. 1–13. DOI: 10.1016/j.compfluid.2018.11.007. URL: https://www.sciencedirect.com/science/article/pii/S0167610518307682.

[69] Hrvoje Jasak, Aleksandar Jemcov, and Zeljko Tukovic. "OpenFOAM: Open source CFD in research and industry". In: *International conference on numerical methods in fluid dynamics*. Springer. 2013, pp. 1–20. DOI: 10.1007/978-3-642-01973-9_1. URL: https://www.researchgate.net/publication/228879492_OpenFOAM_A_C_library_for_complex_physics_simulations.

[70] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. 2nd. Pearson Education, 2007. ISBN: 978-0131274983.

[71] Chao Wang et al. "An Automated Computational Fluid Dynamics Workflow for Simulating the Internal Flow of Race Car Radiators". In: *SAE Technical Paper* 2019-01-0667 (2024). DOI: 10.4271/2019-01-0667. URL: https://www.researchgate.net/publication/385392623_An_Automated_Computational_Fluid_Dynamics_Workflow_for_Simulating_the_Internal_Flow_of_Race_Car_Radiators.

[72] Hrvoje Jasak, Aleksandar Jemcov, and Zeljko Tukovic. "OpenFOAM: A C++ Library for Complex Physics Simulations". In: *International Workshop on Coupled Methods in Numerical Dynamics* (2007), pp. 1–20. URL: https://openfoam.org.

[73] Jiří Blaźek. *Computational Fluid Dynamics: Principles and Applications*. 4th. Elsevier, 2021. ISBN: 978-0-12-819424-3. URL: https://www.sciencedirect.com/book/9780080999951/computational-fluid-dynamics-principles-and-applications.

[74] The OpenFOAM Foundation. *snappyHexMesh User Guide*. Accessed: 2025-08-03. 2023. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-meshing-snappyhexmesh.html.

[75] Wolf-Heinrich Hucho. *Aerodynamics of Road Vehicles*. Oxford: Butterworth-Heinemann, 1998. ISBN: 978-0-7506-5105-3.

[76] Stephen B. Pope. *Turbulent Flows*. Cambridge: Cambridge University Press, 2000. ISBN: 978-0-521-59104-1.

[77] B. E. Launder and D. B. Spalding. "The numerical computation of turbulent flows". In: *Computer Methods in Applied Mechanics and Engineering* 3.2 (1974), pp. 269–289. DOI: 10.1016/0045-7825(74)90029-2.

[78] F. R. Menter. "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications". In: *AIAA Journal* 32.8 (1994), pp. 1598–1605. DOI: 10.2514/3.12149. URL: https://doi.org/10.2514/3.12149.

[79] Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Albuquerque, NM: Hermosa Publishers, 1998. ISBN: 978-0913478080.

[80] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge, UK: Cambridge University Press, 2010. ISBN: 978-0521113601.

[81] H. W. Tieleman and F. T. M. Nieuwstadt. "Drag coefficient of a cube in turbulent flow". In: *Journal of Fluid Mechanics* 228 (1991), pp. 501–510. DOI: 10.1017/S0022112091002680.

[82] A. Larsen. "Drag coefficient of a cube in turbulent flow". In: *Journal of Wind Engineering and Industrial Aerodynamics* 90.14 (2002), pp. 1727–1735. DOI: 10.1016/S0167-6105(02)00094-3.

[83] I. B. Celik, U. Ghia, and P. J. Roache. "Procedure for Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications". In: *Journal of Fluids Engineering* 130.7 (2008), p. 078001. DOI: 10.1115/1.2953232. URL: https://doi.org/10.1115/1.2953232.

[84] P. J. Roache. "A Method for Uniform Reporting of Grid Refinement Studies". In: *Journal of Fluids Engineering* 116.3 (1994), pp. 405–413. DOI: 10.1115/1.2910294. URL: https://doi.org/10.1115/1.2910294.

[85] L. E. Schwer. *IS YOUR MESH REFINED ENOUGH? Estimating Discretization Uncertainty in LS-DYNA Simulations*. Tech. rep. DYNAmore GmbH, 2008. URL: https://www.dynamore.de/de/download/papers/forum08/dokumente/I-I-03.pdf.

[86] T. S. Phillips, C. J. Roy, and I. B. Celik. "Richardson Extrapolation-Based Discretization Uncertainty Estimation for Computational Fluid Dynamics". In: *Journal of Fluids Engineering* 136.12 (2014), p. 121401. DOI: 10.1115/1.4028664. URL: https://doi.org/10.1115/1.4028664.

[87] NASA Glenn Research Center. *Examining Spatial (Grid) Convergence*. 2021. URL: https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html.

[88] J. Ahrens, B. Geveci, and C. Law. "Paraview: An End-User Tool for Large Data Visualization". In: *Visualization Handbook* (2005), pp. 717–731. URL: https://www.paraview.org/paraview.pdf.

[89] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. 3rd. Berlin, Germany: Springer, 2002.

[90] K. Okamoto, M. Nakaoka, and T. Tamura. "Numerical Study on the Recirculation Length in the Wake of a Cube". In: *Computers Fluids* 36 (2007), pp. 1471–1482. DOI: 10.1016/j.compfluid.2007.01.007. URL: https://doi.org/10.1016/j.compfluid.2007.01.007.

[91] T. Tamura and T. Miyagi. "Flow Around a Cube in a Uniform Stream". In: *Journal of Fluids Engineering* 114.2 (1992), pp. 289–294. DOI: 10.1115/1.2910300. URL: https://doi.org/10.1115/1.2910300.

[92] A. Roshko. “On the Development of Turbulent Wakes from Vortex Streets”. In: *National Advisory Committee for Aeronautics, Report* NACA-Report-1191 (1955). URL: https://ntrs.nasa.gov/citations/19930092307.

[93] B. Blocken and T. Stathopoulos. “CFD Simulation of Wind Flow Around Buildings”. In: *Building and Environment* 45.10 (2010), pp. 2233–2240. DOI: 10.1016/j.buildenv.2010.03.004. URL: https://doi.org/10.1016/j.buildenv.2010.03.004.

[94] J. Kim and P. Durbin. “Investigation of Separated Flow Behind a Cube Using Large-Eddy Simulation”. In: *Journal of Fluid Mechanics* 275 (1994), pp. 27–60. DOI: 10.1017/S0022112094002980. URL: https://doi.org/10.1017/S0022112094002980.

[95] S. Saha and S. Menon. “Experimental Study of Symmetry Breaking in Cube Wakes at Moderate Reynolds Numbers”. In: *Experiments in Fluids* 55 (2014), p. 1683. DOI: 10.1007/s00348-014-1683-2. URL: https://doi.org/10.1007/s00348-014-1683-2.

[96] C. Norberg. “Effects of Reynolds number and aspect ratio on the flow around a rectangular cylinder”. In: *Journal of Fluid Mechanics* 258 (1994), pp. 287–316. DOI: 10.1017/S0022112094001371.

[97] C. D. Cantwell and A. Colonius. “Modeling and simulation of forces on bluff bodies in turbulent flows”. In: *Journal of Fluids and Structures* 55 (2015), pp. 329–345. DOI: 10.1016/j.jfluidstructs.2015.03.007.

[98] M. M. Zdravkovich. *Flow Around Circular Cylinders: Volume 1: Fundamentals*. Oxford, UK: Oxford University Press, 1997.

[99] C. J. Baker. “Flow past a cube placed on a plane wall”. In: *Journal of Fluid Mechanics* 92 (1979), pp. 663–673. DOI: 10.1017/S0022112079002586.

[100] J. D. Walker. “Experimental study of the flow past a cube on a moving floor”. In: *Experimental Thermal and Fluid Science* 25 (2002), pp. 23–34. DOI: 10.1016/S0894-1777(01)00141-5.

[101] Maximilian Klank and Michael Beetz. “Probabilistic Framework for Robust Sensor-based Perception in Manipulation Tasks”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5207–5214. DOI: 10.1109/ICRA.2014.6907520.

[102] R. S. Laramee et al. “The State of the Art in Flow Visualization: Dense and Texture-Based Techniques”. In: *Computer Graphics Forum* 23.2 (2004), pp. 203–221. DOI: 10.1111/j.1467-8659.2004.00747.x.

[103] E. J. Stamhuis and J. de Vries. “Streamline Visualization of Flow Patterns in Experimental Fluid Mechanics”. In: *Experiments in Fluids* 11.2-3 (1991), pp. 137–145. DOI: 10.1007/BF00196492.

Appendix A

Ethical Approval Letter

13 June 2025

Dear Mr Denton ,

Reference: CURES/25649/2025
Project ID: 29154
Title: Design and Validation of a Cloud-Based, GPU-Accelerated OpenFOAM Workflow for Automating External-Aerodynamics Simulations - Verification and Validation against experimental results

We are pleased to inform you that you have successfully declared that your research project is a **Literature Review – based solely on openly available literature which is in the public domain, and you are undertaking desk-based research not involving any other form of data or information.**

You have also confirmed that your project **does not meet** any of the literature review specified exceptions, listed both within the relevant section of the CURES form and below:

1) Your supervisor has requested that you apply for approval through CURES because:

- The journals or data are in a sensitive area (please discuss this with your supervisor)
- The project will be embargoed **i.e. will not be publicly available via the Cranfield library immediately or longer term**

2) Approval is/will be specifically required by another external body e.g. journal publishers

Therefore, **you do not require ethical approval** and your CURES application will be automatically closed.

Please keep a copy of this letter safe, if this exception is in relation to your thesis project, you will need to include a copy with your final thesis submission.

If you have any queries, please contact CURES Support.

We wish you every success with your project.

Regards,

CURES Team