

Programa de Doctorado en Ingeniería Matemática,
Estadística e Investigación Operativa

por la

Universidad Complutense de Madrid

y la

Universidad Politécnica de Madrid



Accelerating Computational Fluid Dynamics with Unsupervised Learning-Based Mesh Adaptation

Tesis Doctoral
KENZA Tlales

Director
Dr. GONZALO Rubio Calzado

Año
2025



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros de
Telecomunicación

**Doctoral Degree in Mathematical Engineering, Statistics and
Operations Research**

**Accelerating Computational Fluid Dynamics
with Unsupervised Learning-Based Mesh
Adaptation**

DOCTORAL THESIS

Submitted for the degree of Doctor by:

KENZA Tlales

Master degree in Applied Mathematics

Under the supervision of:
Dr. GONZALO Rubio Calzado

Madrid, 2025

Title: Accelerating Computational Fluid Dynamics with Unsupervised Learning-Based Mesh Adaptation

Author: KENZA Tlaes

Doctoral Programme: Mathematical Engineering, Statistics and Operations Research

Thesis Supervision:

Dr. GONZALO Rubio Calzado, Associate Profesor, ETSIAE UPM(Supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

Abstract

This thesis explores the integration of machine learning techniques into computational fluid dynamics (CFD) to enhance simulation accuracy and computational efficiency. Utilizing the Gaussian Mixture Model (GMM) and unsupervised clustering methods, this research introduces innovative approaches to mesh adaptation and flow feature detection.

The methodology focuses on the application of GMM to partition flow fields into distinct viscous/turbulent and inviscid/irrotational regions. This clustering-driven p -adaptation technique eliminates the need for predefined thresholds, enabling automated, efficient, and accurate mesh refinement. Extensive numerical experiments on a number of test cases, including flow past a cylinder at various Reynolds numbers, demonstrate the ability of the proposed method to balance computational cost and solution fidelity.

Practical engineering applications, such as wind turbine wake simulations, further illustrate the scalability of the proposed methods. The results demonstrate significant improvements in computational efficiency and adaptability, making these techniques well suited for a wide range of CFD applications.

This work contributes to the state-of-the-art by seamlessly integrating machine learning into high-order CFD solvers, paving the way for future advancements in adaptive simulation methods. The proposed methodologies establish a foundation for exploring error-based refinement, three-dimensional turbulence modeling, and the broader adoption of machine learning in fluid dynamics research.

Keywords: Computational fluid dynamics, machine learning, Gaussian Mixture Model,

p -adaptation, flow feature detection, mesh refinement, high-order methods.

Resumen

Esta tesis explora la integración de técnicas de aprendizaje automático en la dinámica de fluidos computacional (CFD, por sus siglas en inglés) para mejorar la precisión de las simulaciones y la eficiencia computacional. Utilizando el Modelo de Mezclas Gausianas (GMM, por sus siglas en inglés) y métodos de agrupamiento no supervisado, esta investigación introduce enfoques innovadores para la adaptación de mallas y la detección de características de flujo.

La metodología se centra en la aplicación del GMM para particionar los campos de flujo en regiones distintas: viscosas/turbulentas e no viscosas/irrotacionales. Esta técnica de p -adaptación impulsada por agrupamiento elimina la necesidad de umbrales predefinidos, permitiendo un refinamiento de malla automatizado, eficiente y preciso. Experimentos numéricos extensos en una serie de casos de prueba, incluyendo el flujo alrededor de un cilindro a diferentes números de Reynolds, demuestran la capacidad del método propuesto para equilibrar el coste computacional y la fidelidad de la solución.

Aplicaciones prácticas de ingeniería, como las simulaciones de estelas de aerogeneradores, ilustran aún más la escalabilidad de los métodos propuestos. Los resultados demuestran mejoras significativas en la eficiencia computacional y la adaptabilidad, lo que convierte a estas técnicas en herramientas adecuadas para una amplia gama de aplicaciones en CFD.

Este trabajo contribuye al estado del arte al integrar de manera fluida el aprendizaje automático en los solucionadores de CFD de alto orden, abriendo el camino a futuros avances en métodos de simulación adaptativa. Las metodologías propuestas establecen una base para explorar la refinación basada en errores, la modelización tridimensional de la turbulencia y la adopción más amplia del aprendizaje automático en la investigación de la dinámica de fluidos.

Keywords: Dinámica de fluidos computacional, aprendizaje automático, Modelo de Mezclas Gaussianas, p -adaptación, detección de características de flujo, refinamiento de mallas, métodos de alto orden.

Résumé

Cette thèse explore l'intégration des techniques d'apprentissage automatique dans la dynamique des fluides numérique (CFD) afin d'améliorer la précision des simulations et la surcharge de calcul. En utilisant le modèle de mélanges gaussiens (GMM) et des méthodes de regroupement non supervisé, cette recherche introduit des approches innovantes pour l'adaptation du maillage et la détection des structures d'écoulement.

La méthodologie se concentre sur l'application du GMM pour partitionner les champs d'écoulement en régions distinctes, différenciant les zones visqueuses/turbulentes des zones inviscides/irrotationnelles. Cette technique de p -adaptation basée sur le regroupement élimine le besoin de seuils prédéfinis, permettant ainsi un raffinement du maillage automatisé, efficace et précis. De nombreuses expériences numériques sur divers cas tests, notamment l'écoulement autour d'un cylindre à différents nombres de Reynolds, démontrent la capacité de la méthode proposée à équilibrer le coût computationnel et la fidélité de la solution.

Des applications en ingénierie, telles que la simulation du sillage d'une éolienne, illustrent davantage l'évolutivité des méthodes proposées. Les résultats montrent des améliorations significatives en termes du coût computationnel et d'adaptabilité, rendant ces techniques particulièrement adaptées à un large éventail d'applications en CFD.

Ce travail contribue à l'état de l'art en intégrant de manière fluide l'apprentissage automatique dans les solveurs CFD d'ordre élevé, ouvrant ainsi la voie à de futures avancées dans les méthodes de simulation adaptative. Les méthodologies proposées établissent une base pour l'exploration du raffinement basé sur l'erreur, la modélisation tridimensionnelle de la turbulence et l'adoption plus large de l'apprentissage automatique dans la recherche en dynamique des fluides.

Mots-clés : Dynamique des fluides numérique, apprentissage automatique, modèle de mélanges gaussiens, p -adaptation, détection des structures d'écoulement, raffinement du maillage, méthodes d'ordre élevé.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Gonzalo Rubio, for his invaluable guidance, support, and encouragement throughout the course of this work. His expertise and dedication have been valuable in shaping this thesis and my academic journey.

I owe a profound debt of gratitude to my beloved parents. To my mother, whose unwavering support and prayers have always been a source of strength for me. To the memory of my late father, whose values and wisdom continue to guide me every day. This accomplishment is a tribute to both of you.

To my wonderful sisters, Dalila, Saida, Rahima, Souheyla, and Nadjat, and to my brothers, Ismail, Elkacem, Abdessalam, and Aissam, thank you for your constant encouragement and love. Your belief in me has been a driving force behind my achievements.

A special acknowledgement goes to my nephews, Sadja, Mouhamad Amir, Mousaab, Youba, Mouhib and Mayar, whose prayers and well-wishes have touched my heart and motivated me to succeed.

To my husband, Youcef, your patience, understanding, and love have meant everything to me. Your unwavering support has been my rock through this challenging journey.

I would also like to extend my heartfelt thanks to my housemates, Chahinez, Sara, and Nourelhouda, with whom I shared countless cherished moments. Together, we were like a family, and I will forever treasure our time together.

To my colleagues, Kheir-Eddine, Andres, Gerasimos, Himpo, Pablo, and all the students in the office I used to work in, thank you for your camaraderie, collaboration, and for making this journey so much more fulfilling.

This work is the result of the collective support and encouragement of many incredible

individuals whose contributions, in various ways, have helped bring it to completion. To each of you, I am deeply grateful.

Finally, I would like to express my gratitude to UPM and the Madrid Supercomputing and Visualization Center (CeSViMa) for providing me with valuable resources.

Contents

Abstract	iii
Resumen	v
Résumé	vii
Acknowledgement	ix
1 Introduction	1
1.1 Motivation	1
1.2 Objectives and Methodolody	4
1.3 Outline of the thesis	6
1.4 Highlights and Novel Contributions	7
I Background on CFD and Machine Learning	9
2 Computational Fluid Dynamics	11
2.1 Introduction to CFD	12
2.2 Navier-Stokes Equations	13
2.3 Methods of Discretization	15
2.4 Mesh Generation and Adaptation	20
2.4.1 Mesh Generation	20
2.4.2 Mesh Adaptation	22
2.5 Conclusion	24
3 Machine Learning	25
3.1 Learning Types in Machine Learning	26
3.1.1 Supervised Learning	26
3.1.2 Unsupervised Learning	26
3.1.3 Semi-supervised Learning	27
3.1.4 Reinforcement Learning	27
3.2 Machine Learning Algorithms	29

3.2.1	Supervised Learning Algorithms	29
3.2.2	Unsupervised Learning Algorithms	33
3.3	Conclusion	36
II	Enhancing CFD by Machine Learning	38
4	Computational and Data-Driven Frameworks	40
4.1	Computational Fluid Dynamics	41
4.1.1	High-order discontinuous Galerkin solver	41
4.1.2	Discontinuous Galerkin Spectral Element Method (DGSEM)	41
4.2	Machine Learning	44
4.2.1	Data Analysis and Clustering with Gaussian Mixture Models: A Python Framework	44
4.2.2	Gaussian Mixture Model	47
4.2.2.1	GMM with EM Algorithm	47
4.2.2.2	Probability Model for Observing Data	48
4.2.2.3	Log-Likelihood and Log-Likelihood Maximization	48
4.2.2.4	Expectation-Maximization (EM) Algorithm Steps	49
4.2.2.5	Jensen's Inequality and Lower Bound	50
4.2.2.6	Constraints in Optimization	51
4.3	Conclusion	51
5	Feature detection using the Gaussian Mixture Model	52
5.1	Methodology and Tools for Feature Detection	53
5.1.1	Feature-based sensors for region separation	53
5.1.1.1	Q-criterion sensor:	53
5.1.1.2	New Ω sensor:	54
5.1.1.3	Gaussian mixture model for flow region detection:	54
5.1.2	Description of test cases	56
5.2	Evaluation and Application of Detection Methods	57
5.2.1	Analysis of the effect of the user-defined threshold	57
5.2.2	Assessment of the quality of the identified regions	60
5.3	Conclusion	63
6	A Clustering-based P-Adaptation Method	72

6.1	Methodology for GMM-Based Mesh Refinement	73
6.2	Results and Discussion	76
6.2.1	Analysis of the effect of the initial mesh resolution	77
6.2.2	Static Vs Dynamic mesh adaptation	79
6.2.3	Static Adaptation Results	81
6.2.3.1	Flow past a cylinder at $Re = 40$	81
6.2.3.2	Flow past a cylinder at $Re = 100$	85
6.2.3.3	Flow past a cylinder at $Re = 3900$	88
6.2.4	Numerical experiments	92
6.3	Conclusion	95
7	Conclusions	99
7.1	Summary of the work	99
7.2	Contributions and Achievements	100
7.3	Future Work	100
7.4	Final Remarks	101
A	Shock Detection using GMM	102
A.1	Description of the Test Cases	103
A.1.1	Inviscid Flow Around a Cylinder at Mach 3	103
A.1.2	Viscous Flow Around a Cylinder at Mach 2	103
A.2	Gaussian Mixture Model (GMM) for Shock Detection	104
A.2.1	Application of GMM to Shock Detection	104
A.2.2	Feature Selection for GMM Clustering	105
A.2.3	Note on the number of clusters for the flow	105
A.3	Comparison with Traditional Shock Sensors	108
A.3.1	Comparison of GMM with Traditional Shock Sensors	108
A.4	Conclusion	112
	Bibliography	125

List of Figures

Figure 3.1	Supervised Learning process, taken from [1]	26
Figure 3.2	The standard reinforcement-learning model, taken from [52]	29
Figure 3.3	Mathematical model of the formal neuron, taken from [101]	30
Figure 3.4	Convolutional Neural Network architecture (CNNs)	32
Figure 3.5	Recurrent Neural Network, taken from [66]	33
Figure 3.6	Autoencoder, taken from [40]	34
Figure 5.1	Flow past a cylinder at $Re = 40$: Flow regions selected using the Q-criterion sensor f , $K = -0.453$ (a), $K = -0.4909$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.	64
Figure 5.2	Flow past a cylinder at $Re = 100$: Flow regions selected using the Q-criterion sensor f , $K = -0.193$ (a), $K = 0.343$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.	65
Figure 5.3	Flow past a cylinder at $Re = 3900$: Flow regions selected using the Q-criterion sensor f , $K = -0.4679$ (a), $K = 0.4014$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Regions Viscous rotational region, Blue: Inviscid irrotational region.	66
Figure 5.4	Flow past a cylinder at $Re = 40$: Flow regions selected using the Ω sensor g , $K = 0.1$ (a), $K = 0.0156$ - automatic threshold computed with GMM. (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.	67

Figure 5.5 Flow past a cylinder at $Re = 100$: Flow regions selected using the Ω sensor g , $K = 0.04625$ (a), $K = 0.0013$ - automatic threshold computed with GMM. (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red : Viscous region, Blue: Inviscid region.	68
Figure 5.6 Flow past a cylinder at $Re = 3900$: Flow regions selected using the Ω sensor g , $K = 0.03278$ (a), $K = 0.0003$ - automatic threshold computed with GMM (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E (d). Red : Viscous rotational region, Blue: Inviscid irrotational region.	69
Figure 5.7 Scatter map $(Q_\Omega, -Q_S)$ in the inviscid region for the flow past a cylinder at $Re = 40$	70
Figure 5.8 Scatter map $(Q_\Omega, -Q_S)$ in the inviscid region for the flow past a cylinder at $Re = 100$	70
Figure 5.9 Scatter maps $(Q_\Omega, -Q_S)$, $(R_S, -Q_S)$ and (Q_Ω, R_S) in the inviscid regions for the flow past a cylinder at $Re = 3900$	71
Figure 6.1 Clustering and regions marked for adaptation when considering various polynomial orders to train the clustering GMM algorithm ($P_{initial} \in [2, 4]$ for $Re = 40$, $Re = 100$ and $P_{initial} \in [3, 4]$ for $Re = 3900$). Flow past a cylinder at $Re = 40$ (6.1a), $Re = 100$ (6.1b) and $Re = 3900$ (6.1c). Red line : Boundary of the viscous/turbulent region detected with GMM using $P = 4$. Dashed blue line : Boundary of the viscous/turbulent region detected with GMM using $P = 2$ for the flow at $Re = 40$, $Re = 100$ and $P = 3$ for the flow at $Re = 3900$. Dark Grey : Elements marked for adaptation when using either the red or blue clustered regions. Light Grey : Elements marked for adaptation which are selected only by one of the clustered regions (no overlapping).	78
Figure 6.2 Flow past a cylinder at $Re = 3900$: Flow region clustering at different time instants using GMM clustering with feature space E	80
Figure 6.3 Figure 6.3a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 40$. Grey : $P_{cluster} = 4$, White : $P_{inviscid} \in [1, 2, 3]$, Red line : Boundary of the viscous dominated, rotational region detected with GMM. Figure 6.3b presents the norm of velocity for the adapted flow field.	82

Figure 6.4	Max residuals (the infinity norm of the temporal derivative in the semi-discrete Navier-Stokes equations) for homogeneous and p -adapted solutions	82
Figure 6.5	Comparison of drag coefficients for flow past a cylinder at $Re = 40$. Individual data points represent uniform solutions at $P = 2$, $P = 3$, and $P = 4$, along with the GMM clustering-based technique employing $P_{cluster} = 4$ and $P_{inviscid} = 1$. The blue and orange lines illustrate solutions obtained with $P_{cluster} = 4$ and $P_{inviscid} = 1$, utilizing varying thresholds for the Q-criterion and new Ω method, respectively	85
Figure 6.6	Figure 6.6a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 100$. Grey: $P_{cluster} = 4$, White: $P_{inviscid} \in [1, 2, 3]$, Red line: Boundary of the viscous dominated, rotational region detected with GMM. Figure 6.6b presents the norm of velocity contour for the adapted case.	86
Figure 6.7	Figure 6.7a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 3900$. Grey: $P_{cluster} = 4$, White: $P_{inviscid} = 2$, Red line: Boundary of the viscous dominated, rotational/turbulent region detected with GMM. Figure 6.7b presents the contour of the spanwise velocity colored with the axial velocity u for the adapted solution.	89
Figure 6.8	Statistical quantities of the wake for the unsteady turbulent case of a flow past a cylinder at $Re = 3900$. Comparison of experimental (PIV) and high resolution numerical (LES) results from [87] against uniform and adapted results from HORSES3D	97
Figure 6.9	Contours of the Q-criterion, with colors representing the L2 norm of the velocity, for the flow past a wind turbine at $Re_c = 103600$. Figure 6.9a also includes slices of the detected viscous dominated, rotational/turbulent region, highlighted in red, using the Gaussian Mixture Model (GMM) analysis. Figure 6.9b displays both horizontal and vertical planes. The intersection of these planes marks the positions where the velocity deficit at the horizontal axis is compared with experimental data.	98

Figure 6.10 Horizontal profiles of mean streamwise velocity deficit at three downstream positions x/D for $P \in [1, 2, 3, 4]$ and adapted ($P_{cluster} = 4$ and $P_{inviscid} = 1$) and comparison with experimental data [61].	98
Figure A.1 Bayesian information criterion with respect to the number of clusters.	106
Figure A.2 Feature-space clustering with the GMM sensor applied to the viscous case.	107
Figure A.3 Viscous case after 300,000 iterations with the modal sensor. . . .	109
Figure A.4 Viscous case after 300,000 iterations with the integral sensor. . .	110
Figure A.5 Viscous case after 300,000 iterations with our adaptive GMM sensor using 6 clusters. The simulation with the sensor based on $(\nabla \cdot v)^2$ was restarted from the 10,000th time step of the case using $\ \nabla p\ ^2, (\nabla \cdot v)^2$	111

List of Tables

Table 6.1 Comparison between static and dynamic adaptation for the flow past a cylinder at $Re = 100$	80
Table 6.2 Comparison of the degrees of freedom (DoF), computational cost and drag coefficient (C_d) for the uniform and adapted solutions for the steady flow past a circular cylinder at $Re = 40$	83
Table 6.3 Comparison of the drag coefficient (C_d) results for flow past a cylinder at $Re = 40$ against experimental data	83
Table 6.4 Average drag coefficient, amplitude of the lift coefficient oscillations and Strouhal number results for the unsteady flow past a circular cylinder at $Re = 100$	87
Table 6.5 Comparison of the degrees of freedom (DoF) and computational cost for the uniform and adapted solutions for the unsteady flow past a circular cylinder at $Re = 100$	88
Table 6.6 Comparison of numerical results from the literature for the unsteady flow past a circular cylinder at $Re = 100$	88

Table 6.7	Comparison of the mean statistical quantities for the cylinder at $Re = 3900$ using HORSES3D and literature data. We compare the Strouhal number St , the drag coefficient C_d , and the recirculation length L_r . All simulations used a spanwise length $L_z/D = \pi$.	90
Table 6.8	Comparison of DoFs and computational time for the cylinder at $Re = 3900$	91
Table 6.9	Comparison of DoFs and computational time for the turbine test case.	95
Table A.1	AIC and BIC values presented in Figure A.1.	106
Table A.2	Comparison of computational cost for different shock sensors in the viscous cylinder flow simulation. Values represent the percentage of total computational resources used.	112

Chapter 1

Introduction

1.1. Motivation

Computational fluid dynamics (CFD) has become an essential tool across many scientific and engineering fields, including aerospace, automotive, and environmental engineering. This field of study relies on solving the partial differential equations (PDEs) that govern fluid flow behavior, such as the Navier-Stokes equations, to obtain accurate numerical simulations of fluid dynamics. CFD simulations are critical in both academia and industry as they allow for the detailed investigation of complex flow phenomena that would be costly or impractical to explore experimentally [51, 31]. However, CFD simulations face a major challenge: balancing computational cost with the level of accuracy required. To resolve intricate flow details, such as boundary layers, shock waves, and turbulent structures, simulations often need highly resolved computational grids, which increase computational demand. This has driven significant research into adaptive mesh refinement techniques that can focus computational resources only where they are needed, enhancing the efficiency and feasibility of high-fidelity simulations [50].

A major challenge in CFD lies in achieving a compromise between computational accuracy and efficiency. High-resolution grids offer better accuracy in critical regions of the flow field, such as near boundaries or in regions with steep gradients. However, creating high-resolution grids across the entire computational domain can be prohibitively costly, making it inefficient for practical applications [118]. To address this, localized refinement strategies have been developed to allocate higher grid reso-

lution only where necessary, such as near boundary layers or around shock regions, while maintaining coarser grids in areas where the flow field is relatively uniform. This approach allows for a more balanced distribution of computational resources, maximizing efficiency without compromising on accuracy. However, existing adaptive techniques also face substantial limitations, primarily because they rely on manually defined criteria and error estimates that are not always robust or universally applicable across diverse flow conditions [4, 115].

To achieve efficient simulations, various mesh adaptation techniques, specifically h -adaptation and p -adaptation, have been widely studied. h -adaptation techniques focus on refining the mesh by adding more elements in regions where higher accuracy is needed. For example, h -adaptation is frequently used to increase element density near boundary layers, where steep velocity and pressure gradients require a fine resolution to accurately capture flow characteristics [3]. This approach effectively improves accuracy in high-gradient areas but comes at a computational cost, as the increased number of elements leads to a greater computational load. Additionally, h -adaptation relies on error estimators or gradient-based criteria, often requiring significant user input to specify thresholds for refinement. This dependence on user-defined parameters presents a challenge for generalization, as the chosen thresholds may not perform consistently across different flow regimes [36].

Complementing h -adaptation, p -adaptation techniques adapt the polynomial degree of basis functions within each element rather than increasing the mesh density. Higher polynomial degrees can capture flow characteristics with fewer elements, which can be advantageous in terms of computational resources. However, determining the appropriate polynomial degree for each element is nontrivial and typically depends on error estimates or heuristic criteria. This adds another layer of complexity, as inappropriate polynomial orders can lead to over- or under-refinement, impacting both accuracy and efficiency [43]. As with h -adaptation, p -adaptation methods often require extensive user-defined parameters, which can limit their robustness across varied flow conditions. Furthermore, both h - and p -adaptation require repeated refinement cycles to converge on a solution, which further increases computational costs, particularly in unsteady flow simulations where regions of interest change dynamically [116].

Error estimation plays a significant role in adaptive mesh refinement, guiding re-

finement processes by identifying areas of high error that may require finer mesh resolution or higher polynomial degrees. Common error indicators include residual-based and interpolation-error-based methods, each with advantages and limitations. Residual-based indicators, for instance, are valuable for their ability to provide local error information, but they often require multilevel solutions to be effective, which adds to the computational cost [27]. Interpolation-error indicators, on the other hand, are more direct but may yield inaccurate results if they are not carefully calibrated for each specific problem, often resulting in either under- or over-refinement in particular regions of the flow [20]. These challenges highlight the inherent limitations in existing adaptation techniques and underscore the need for more efficient, automated approaches that reduce reliance on manual parameter tuning [82].

The difficulties associated with manually tuning refinement criteria expose a significant gap in traditional CFD approaches. Adaptive methods that rely on user-defined parameters are often not well suited for simulations where the flow characteristics vary greatly within the domain or over time, as seen in applications like turbulence modeling or high-speed aerodynamic simulations. In many cases, determining suitable refinement thresholds is a labor-intensive process, requiring extensive trial and error to achieve the desired balance between computational efficiency and accuracy. Consequently, the rigidity of current adaptation techniques limits their applicability in dynamically changing environments or complex flows, such as those around bluff bodies or in multiphase flows [19].

Another notable challenge in adaptive mesh refinement is identifying the regions that require refinement without excessive manual intervention. This is especially true in unstructured grids, where the adaptation process involves modifying the grid topology in a way that ensures stability and maintains accuracy [39]. These limitations emphasize the importance of a more automated, data-driven adaptation framework that can dynamically adjust to changing flow conditions and complexities, improving robustness, and reducing computational expense.

Given the limitations of traditional approaches, this thesis explores machine learning (ML) as an innovative solution to improve adaptation techniques in CFD. The application of ML has gained significant momentum in computational sciences due to its potential to uncover data-driven insights and automate processes that would otherwise require extensive manual tuning. Specifically, ML holds promise in au-

tomating mesh refinement and polynomial adaptation by identifying regions that require refinement based on data patterns, rather than relying on predefined criteria. This could substantially reduce the need for manual intervention, making adaptation techniques more robust and adaptable to complex flow scenarios [12, 93]. Recent advances in ML have shown its potential to transform CFD methodologies. Supervised ML approaches have been successfully applied to detect flow features, such as turbulent regions and boundary layers, as demonstrated by [64] and [102]. Neural networks, for example, have been used to predict element aspect ratios for h -adaptation [33] and for goal-oriented mesh refinement [120]. However, supervised approaches often require extensive labeled datasets, limiting their applicability to diverse or evolving flow conditions.

Unsupervised ML offers significant potential in CFD for addressing challenges that traditional methods and supervised approaches struggle to overcome. Clustering techniques, such as Gaussian Mixture Models (GMMs), are particularly well suited to detect regions that require refinement based on inherent patterns in the flow data. These algorithms can dynamically identify areas of high gradients or turbulence without requiring explicit labels, making them ideal for adaptive mesh refinement, especially in tasks where flow characteristics are not known in advance [84]. Using GMM, this thesis proposes a novel approach to detect regions that need refinement, such as viscous or turbulent zones, thereby eliminating the need for manual thresholds and reducing dependency on user-defined parameters. Unlike traditional methods, which rely on predefined criteria to differentiate between regions, the ML-based approach can dynamically adjust to variations within the flow field, leading to more efficient and accurate simulations.

1.2. Objectives and Methodology

The primary objective of this thesis is to enhance Computational Fluid Dynamics (CFD) simulations through the integration of unsupervised machine learning techniques. Specifically, the research focuses on improving the accuracy and efficiency of simulations by optimizing mesh refinement processes and enhancing p -adaptation techniques. This approach addresses critical challenges in traditional CFD methods, such as reliance on predefined thresholds and manual effort required to detect regions

of high gradients or turbulent flow.

Traditional mesh adaptation techniques rely heavily on user-defined parameters, which are often insufficient for complex, dynamically evolving flow scenarios. In contrast, unsupervised ML provides a data-driven alternative that identifies areas requiring refinement or shock capture based on inherent patterns in flow data, eliminating the need for prior knowledge or manual tuning. This work aims to explore these innovative approaches in the context of advancing p -adaptation and shock detection methods, drawing inspiration from the potential of high-order methods outlined in [121] and the future directions highlighted in the “CFD Vision 2030 Study” [108].

Recent advancements in ML show its potential to transform CFD methodologies. Supervised ML has been used to detect flow features like turbulent regions and boundary layers, as seen in [64] and [102]. However, these approaches require extensive labeled datasets, limiting their application. In contrast, unsupervised ML, though underexplored, offers the ability to generalize across untrained scenarios. Clustering techniques like Gaussian Mixture Models (GMMs) can detect high-gradient or turbulent zones for adaptive mesh refinement, addressing long-standing challenges in CFD.

Machine learning (ML) offers promising solutions for computational fluid dynamics (CFD). While supervised methods effectively detect flow features [64, 102] and aid in mesh refinement [33, 120], they require extensive labeled data. In contrast, unsupervised approaches—such as Gaussian Mixture Models (GMMs) [84]—automatically identify critical flow regions without labeled datasets, offering greater adaptability. This thesis uses these unsupervised techniques to address persistent CFD challenges.

To address these challenges, this work is structured into several key components to address the challenges of adaptive refinement in Computational Fluid Dynamics (CFD) to enhance the accuracy and efficiency of simulations, particularly in regions with complex flow features. First, CFD data will be generated using the high-order spectral element CFD solver HORSES3D [29] to create a diverse dataset covering a wide range of flow scenarios. The dataset will include steady-state and unsteady simulations, providing a comprehensive basis for identifying patterns relevant to flow complexity.

Next, unsupervised learning techniques, such as clustering algorithms, will be employed to analyze the flow data. Specifically, methods such as Gaussian mixture models

(GMMs) will be used to identify regions of interest, such as areas with high velocity gradients or turbulent features. These regions will be targeted for refinement, thereby eliminating the need for manually defined thresholds.

The insights derived from unsupervised machine learning will then guide p -adaptation, adjusting polynomial orders locally to enhance solution accuracy. Unlike traditional methods that rely on heuristics or predefined error metrics, this data-driven approach automatically adapts to flow characteristics, reducing dependence on user-defined parameters, ensuring a more objective and adaptable method to mesh adaptation, aligning with objectives outlined by previous research such as that of [121].

Finally, the effectiveness of the proposed framework will be validated against a set of benchmark problems. Metrics such as solution accuracy, computational efficiency, and robustness across various flow conditions will be analyzed to quantify the improvements introduced by the ML-guided methodology.

Using unsupervised machine learning, this approach addresses long-standing challenges in CFD, such as automating feature detection and mesh refinement while reducing the need for manual intervention. The integration of ML into CFD workflows represents a significant step toward more intelligent and efficient simulation techniques.

1.3. Outline of the thesis

- Part I: Foundational Knowledge in Computational Fluid Dynamics (CFD). This part establishes the theoretical and methodological groundwork for the thesis. Key topics include:
 - Navier-Stokes Equations: An explanation of the fundamental equations governing fluid dynamics and their relevance in various flow scenarios.
 - Numerical Methods and Discretization: Discussion of discretization methods, mesh generation, and adaptation techniques, essential for solving the Navier-Stokes equations.
 - Advanced Numerical Methods: Introduction to the Discontinuous Galerkin

Spectral Element Method (DGSEM) and its advantages in achieving higher resolution and accuracy in simulations.

- Integration of Machine Learning in CFD: Overview of how machine learning is being integrated into CFD workflows, setting the stage for the practical applications discussed in Part II.
- Part II: Practical Application of Machine Learning in CFD. This section focuses on the implementation and evaluation of a specific machine learning technique to enhance CFD simulations:
 - Clustering-Based p -Adaptation Methodology: This technique refines computational models by dynamically adjusting the polynomial order within mesh elements based on clustering analysis. Its application is demonstrated across various Reynolds number cases, showcasing its ability to improve simulation efficiency and accuracy.
 - Case Studies and Evaluation: Discussion of practical examples where our technique is applied, including a thorough analysis of results, challenges, and their implications for advancing computational fluid dynamics.

1.4. Highlights and Novel Contributions

This section presents the scientific contributions made during the development of this thesis, including journal publications, conference presentations, and workshops.

Journal Publications

The following peer-reviewed journal articles were published as part of this research:

- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Gonzalo Rubio, Esteban Ferrer. Machine learning mesh-adaptation for laminar and turbulent flows: applications to high-order discontinuous Galerkin solvers. *Engineering with Computers*, vol. 40, no. 5, pp. 2947–2969, 2024. Springer.
- Andrés Mateo-Gabín, **Kenza Tlales**, Eusebio Valero, Esteban Ferrer, Gonzalo Rubio. An unsupervised machine-learning-based shock sensor: Application to

high-order supersonic flow solvers. *Expert Systems with Applications*, p. 126352, 2025. Elsevier.

Conference, Seminar, and Workshop Presentations

- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Gonzalo Rubio, Esteban Ferrer. *Machine Learning Mesh Adaptation for Laminar and Turbulent Flows: Applications to High-Order Discontinuous Galerkin Methods*. Workshop Junior Interdisciplinar, Madrid, Spain (IMEIO), 2023
- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Andrés Mateo-Gabín, Gonzalo Rubio, Esteban Ferrer. *Enhancing High-Order Discontinuous Galerkin Solvers through Machine Learning-Based Mesh Adaptation and Machine learning-based Shock Capturing*. Annual Research Plan Presentation, Universidad Politécnica de Madrid (UPM), 2023
- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Gonzalo Rubio, Esteban Ferrer. *Adaptive Mesh Strategies for Laminar and Turbulent Flows using Machine Learning*. Numath's Seminar, Universidad Politécnica de Madrid (UPM), 2023
- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Gonzalo Rubio, Esteban Ferrer. *Machine Learning-Driven Mesh Refinement for High-Order Fluid Simulations*. 2nd Spanish Fluid Mechanics Conference, Universitat Politècnica de Catalunya, Barcelona, Spain, 2023
- **Kenza Tlales**, Kheir-Eddine Otmani, Gerasimos Ntoukas, Andrés Mateo-Gabín, Gonzalo Rubio, Esteban Ferrer. *Unsupervised Learning Strategies for Mesh Adaptation and Shock Capturing in High-Order CFD Solvers*. III Workshop Junior Interdisciplinar, Madrid, Spain (IMEIO), 2024

Training: Modeling Weeks and Study Groups with Enterprises

- Modelling Week: Machine Learning for Predicting Healthcare Emergency Admissions based on Weather Conditions. Universidad Complutense de Madrid (UCM), Madrid, Spain, 2023

Part I.

Background on CFD and Machine Learning

Summary:

In this part, we introduce the fundamental concepts of Computational Fluid Dynamics (CFD) and Machine Learning (ML) that form the basis of this research. We first present the mathematical and numerical foundations of CFD, including the governing equations and discretization methods. We then provide an overview of machine learning techniques, emphasizing unsupervised learning algorithms such as clustering. These foundational topics establish the necessary background for integrating data-driven approaches into CFD simulations, as developed in the subsequent chapters.

Chapter 2

Computational Fluid Dynamics

Most of the mathematical models of science and engineering take the form of differential or integral equations. The use of these models has increased with the rapid development of high-speed computers over the past 20 years. This allows one to study the most complicated mathematical models in science and engineering, which reduces the time and cost of experimental tests and enables the comparison of many results for improvement. This led to increased activity in Design engineering and Computer-aided manufacturing, and as a result, the emergence of scientific computing as a complement to theoretical and experimental sciences.

This Chapter provides a general overview of Computational Fluid Dynamics (CFD), focusing on its fundamental concepts and techniques. Section 2.1 introduces CFD as a tool for analyzing fluid flow, followed by Section 2.2 (Navier-Stokes Equations), which outlines the equations governing fluid motion. Section 2.3 discusses both low-order methods, such as finite differences and finite volumes, and high-order methods, like the Discontinuous Galerkin method, while Section 2.4 (Mesh Generation and Adaptation) highlights the basics of creating and adapting computational grids for simulations. Through these sections, Chapter 2 lays the groundwork for understanding CFD's complexities and its integration with machine learning approaches in subsequent chapters.

2.1. Introduction to CFD

CFD stands for Computational Fluid Dynamics, a branch of fluid mechanics that relies on computers to produce quantitative predictions of fluid flow phenomena. Fluid flow motion is governed by a set of conservation laws: the so-called Navier–Stokes equations. They were derived in the mid-18th century by the French engineer Claude Navier and the Irish mathematician George Stokes. The continuity, momentum, and energy equations represent the conservation laws of fluid motion.

CFD rewrites the Navier–Stokes equations as a system of algebraic equations that relates velocity, temperature, pressure, etc. and a number of collocation points in a finite-dimensional space. These equations are then integrated numerically, and this gives the most relevant attributes to the flow. The simulation of fluid motion by means of a CFD code has, in general, three main steps: pre-processor to put the problem in the right form (mesh generation, etc), solver which is the program that calculates the solution of the CFD problem (solving the governing equations), and the post-processor which is the step where the results are displayed (plot the solution, averaged quantities, etc), and as a result, the flow will be approximately described [65].

The nature of the flow is related to the Reynolds number (Re), which is a non-dimensional number that specifies the fluid flow's nature. Typically, low Re number simulations are laminar and high Re number problems are turbulent. Turbulence is characterized by its irregular, three-dimensional, and spread nature which we find in most natural phenomena [65].

The most relevant quantities in engineering applications are often the integral or mean quantities of fluid flow, which are analyzed through the application of turbulence modeling. Turbulence resolution can be classified into three levels: fully resolved, partially resolved, and unresolved. Direct Numerical Simulation (DNS), a fully resolved approach, directly solves the Navier–Stokes equations. While DNS is effective for simple problems with low Reynolds numbers, its significant computational cost makes it impractical for high Reynolds number flows [123]. For more complex problems, implicit or explicit Large Eddy Simulation (LES) and Reynolds-Averaged Navier–Stokes (RANS) methods are preferred due to their efficiency in handling high Reynolds number conditions.

Unlike the large scales that depend on specific flow conditions and geometries, the

small scales of turbulence are more homogeneous, which makes them easier to model. Large Eddy Simulation (LES) is based on the idea of spatially filtering out and modeling the small scales of turbulence (close to Kolmogorov scales). While small scales dissipate energy, the large scales are resolved directly by the governing equations. This approach significantly reduces computational cost compared to DNS, but at the expense of increased modeling and algorithmic complexity [126].

The third level of turbulence resolution is addressed by the Reynolds-Averaged Navier–Stokes (RANS) model, which is based on the time-averaged solution to the Navier–Stokes equations. This approach costs significantly less than DNS or LES, making it a practical choice for many engineering applications, despite its limitations in capturing detailed turbulent structures [17].

In the early 1970, these methods appeared, and a decade later, they had a place in actual applications, as the first commercial program was introduced in the early 1980s. Since that time, the concepts of CFD have developed in terms of geometric flexibility, accuracy, efficiency and model complexity, such as multi-phase flow or chemical reactions.

Nowadays, many companies rely on CFD for design and engineering analysis because it helps to predict the performance of new designs or processes before manufacturing.

2.2. Navier-Stokes Equations

The Navier-Stokes (NS) equations are a system of non-linear partial differential equations that describe the motion of fluid flows. These equations are derived from the fundamental conservation laws of mass, momentum, and energy [42]. The three-dimensional compressible Navier-Stokes equations, in conservative form [81], are expressed as:

$$u_t + \nabla \cdot \vec{F}_e = \nabla \cdot \vec{F}_{v,\text{turb}}, \quad (2.1)$$

where u is the conservative variables vector field:

$$u = [\rho, \rho v_1, \rho v_2, \rho v_3, \rho e]^T,$$

and $\vec{F}_e$ represents the inviscid (Euler) flux vector, defined as:

$$\vec{F}_e = \begin{bmatrix} \rho v_1 & \rho v_2 & \rho v_3 \\ \rho v_1^2 + p & \rho v_1 v_2 & \rho v_1 v_3 \\ \rho v_1 v_2 & \rho v_2^2 + p & \rho v_2 v_3 \\ \rho v_1 v_3 & \rho v_2 v_3 & \rho v_3^2 + p \\ \rho v_1 H & \rho v_2 H & \rho v_3 H \end{bmatrix},$$

where ρ is the density, e is the total energy, p is the pressure, the velocity vector is $\vec{v} = [v_1, v_2, v_3]^T$, and the total enthalpy H is given by $H = E + p/\rho$.

The viscous and turbulent fluxes are given by $\vec{F}_{v,\text{turb}}$:

$$\vec{F}_{v,\text{turb}} = \begin{bmatrix} 0 & 0 & 0 \\ \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \\ \sum_{j=1}^3 v_j \tau_{1j} + \kappa T_x & \sum_{j=1}^3 v_j \tau_{2j} + \kappa T_y & \sum_{j=1}^3 v_j \tau_{3j} + \kappa T_z \end{bmatrix},$$

where κ is the thermal conductivity, and T_x, T_y, T_z are the components of the temperature gradient. The stress tensor τ is given by:

$$\tau = (\mu + \mu_t)(\nabla \vec{v} + (\nabla \vec{v})^T) - \frac{2}{3}(\mu + \mu_t)\mathbf{I}\nabla \cdot \vec{v}, \quad (2.2)$$

where μ is the dynamic viscosity and μ_t is the turbulent viscosity (in this work defined through the Vreman model). $\mathbf{I}$ is the three-dimensional identity matrix, and the dynamic turbulent viscosity using the Vreman model [119] is given by:

$$\mu_t = C_v \rho \sqrt{\frac{B_\beta}{\alpha_{ij}\alpha_{ij}}}, \quad \alpha_{ij} = \frac{\partial u_j}{\partial x_i}, \quad \beta_{ij} = \Delta^2 \alpha_{mi} \alpha_{mj}, \quad \Delta = \frac{V^{1/3}}{N+1}, \quad (2.3)$$

$$B_\beta = \beta_{11}\beta_{22} - \beta_{12}^2 + \beta_{11}\beta_{33} - \beta_{13}^2 + \beta_{22}\beta_{33} - \beta_{23}^2,$$

where $C_v = 0.07$ is the model constant, V is the volume of the element, and N is the polynomial order of the approximation. The Vreman model adapts the turbulent viscosity based on the local flow characteristics and automatically reduces the turbulent viscosity in laminar, transitional, and near-wall regions, preserving physical fidelity in compressible LES.

The Smagorinsky model [109] for turbulent viscosity is defined as:

$$\mu_t = (\rho C_s \Delta)^2 \sqrt{2S_{ij}S_{ij}}, \quad (2.4)$$

where $C_s = 0.2$ is the Smagorinsky constant, and $\Delta = V^{1/3}/(P+1)$ is the grid filter (with V the cell volume and P the polynomial order). The rate-of-strain tensor is given by:

$$S = \frac{1}{2} (\nabla \vec{v} + (\nabla \vec{v})^T). \quad (2.5)$$

Equation 2.1 represents the compact form of the continuity, momentum, and energy equations derived from conservation laws. In u , $\vec{F}_e$, and $\vec{F}_{v,\text{turb}}$, the first components define the continuity equation, the next three represent the momentum equation in the directions x , y , and z , and the last component corresponds to energy.

CFD techniques often simplify the full NS equations to reduce this complexity. Some typical simplifications include:

- **Incompressible flows:** $\nabla \cdot \vec{v} = 0$, with constant density ρ .
- **Steady flows:** assuming that the flow quantities do not change over time [6].
- **Inviscid flows:** neglecting viscous terms at high Reynolds numbers ($Re = \rho U_\infty L / \mu$), where U_∞ is the characteristic velocity of the fluid and L is the characteristic length [48]. These are governed by the Euler equations, a special case where $\nabla \cdot \vec{F}_{v,\text{turb}} = 0$.

2.3. Methods of Discretization

The sets of PDEs typically found in CFD do not have analytical solutions, except for very simple cases. Therefore, they must be numerically integrated. Flow domains are split into smaller subdomains where the governing equations are solved. Different geometric shapes such as triangles and rectangles in 2D and tetrahedrons and hexahedrons in 3D are used to form the new domain. Each of these portions of the domain

is known as cell or element. The combination of all elements generate what we call mesh or grid, which is obtained by a process termed mesh/grid generation, and it is considered as an important step in the analysis operation.

Methods of discretization can be classified into low order methods and high order methods:

1. Low Order Methods:

- a) **Finite Difference Method** The finite difference method (FD) has been a foundational approach for solving partial differential equations (PDEs) since its inception in the early 20th century. Contemporary research increasingly focuses on complex nonlinear systems, such as those modeling the Earth's atmosphere or ocean circulation. Analyzing linear model problems instead of nonlinear systems allows for a deeper understanding of the theoretical behavior underlying these complex schemes. The finite difference method is one of the simplest numerical approximation techniques; however, it is important to note that this method is not inherently conservative. In FD, variables are represented by a mesh of discrete points, and the solution is evaluated only at cell interfaces (nodes), with accuracy directly related to the mesh size: finer grids yield more accurate results. The FD method operates by transforming the differential form of the PDE into a system of algebraic equations (discretization). One of the common techniques for deriving finite difference expressions is the Taylor series expansion. Typically, the finite difference method is applied to simple geometries, as imposing boundary conditions on curved domains can be challenging, yet its straightforward implementation contributes to its widespread use in computational fluid dynamics [75].
- b) **Finite Volume Method** The finite volume method divides the physical domain into very small finite-sized elements called cells. Conservation laws are the foundation of the finite volume method: what enters a cell is equal to what exits from the other side, ensuring that flux is conserved in each subdomain. This method is particularly effective for handling nonlinear problems because it requires only the evaluation of flux at the cell interfaces, making it a natural choice for CFD problems. However, compared to the finite difference method, it is more challenging to implement [75]. The

accuracy of this method can be increased through mesh refinement, but it is not straightforward to raise the polynomial order used to approximate the solution within the cell, which is a disadvantage of the finite volume method compared with the finite element method (FEM) [28].

- c) **Finite Element Method** The finite element method consists of subdividing the physical domain into small finite computational domains. The solution is approximated by a simple function, such as a linear or quadratic polynomial, with finite degrees of freedom (DOFs). Unlike the finite difference method, which uses the differential form of the PDE, the starting point of the finite element method is the integral form of the PDE. This general method provides a similar order of accuracy to that obtained from the finite difference (FD) and finite volume (FV) methods. One advantage of the finite element method is its ability to increase the order of the element, which enhances the accuracy of the approximated solution. Additionally, it can effectively handle irregular CAD ¹ geometries. However, it is an advanced method that requires mathematical expertise to implement [75]. While the finite element method is often categorized as a low-order method, it can be formulated as a high-order method by employing higher-degree polynomials for approximation. This flexibility allows for improved accuracy and convergence rates, particularly in problems where smooth solutions are anticipated. With this understanding of low-order methods, we can now explore the characteristics and advantages of high-order methods.

2. High Order Methods

High order numerical methods provide better accuracy for the same number of DOF than low order methods. That is why they are popular in computational fluid dynamics (CFD) to effectively resolve complex-flow features.

a) **Continuous Galerkin method (CG):**

The Continuous Galerkin (CG) method is a widely used finite element technique for numerically solving partial differential equations (PDEs). In CG methods, the solution domain is discretized into finite elements, and the solution is approximated using basis functions that are continuous across element boundaries. This continuity ensures that the approximate

¹CAD: computer-aided design

solution maintains a smooth transition between adjacent elements, which is particularly advantageous for problems where the solution is expected to be smooth. A significant advantage of CG methods is their computational efficiency. The continuity of basis functions across elements leads to a smaller system of equations compared to some other methods, which can result in reduced computational costs. Additionally, CG methods have been successfully applied to a wide range of problems, including structural analysis, fluid dynamics, and heat transfer.

However, CG methods can face challenges when dealing with problems that involve discontinuities or sharp gradients. The enforced continuity across elements can lead to numerical oscillations or inaccuracies in such cases. To mitigate these issues, various stabilization techniques have been developed. For instance, the Discontinuous Galerkin methods (DG) [125]. While both CG and DG methods are used to solve PDEs, they differ in their treatment of solution continuity, computational requirements, and suitability for handling complex solution features. The choice between CG and DG methods depends on the specific problem characteristics and computational considerations.

- b) **Discontinuous Galerkin methods** (DG methods) are a class of high-order methods based on the variational projection of the governing equations, combining features of FV and FE methods. They were first introduced in 1973 by Reed and Hill to numerically solve the neutron transport problem. DG methods are widely used in various industries and branches of science, such as biology [104] and fluid dynamics.

DG methods can be interpreted as FEM but with alleviated constraints on element interfaces. They are high-order methods, where the solution within each element is approximated by a series of orthogonal polynomials of arbitrary degree. Additionally, DG methods are robust, locally conservative and capable of naturally handling discontinuities due to the disconnected nature of domain discretization. They can easily handle elements of arbitrary shapes and irregular triangulations with hanging nodes [16].

One significant advantage of DG methods is that each element can be

treated as a separate entity that requires only boundary data from its neighbours. This characteristic facilitates local refinement, allowing for the treatment of each element as an independent part. Consequently, these methods provide the potential for significantly reducing the computational cost of simulations by increasing local spatial resolution through either locally reducing the mesh size (h-refinement) or locally increasing the polynomial order of approximation within the elements (p-refinement). This approach, known as mesh adaptation, results in a reduction in the number of degrees of freedom (DoF) while maintaining accuracy [57].

- c) **Essentially Non-Oscillatory (ENO) schemes** were introduced in 1987 by Harten et al. [41]. ENO methods are essentially destined to deal with shocks and complicated smooth flow regimes. An interesting property of these methods is that they have been proven to strongly obviate oscillations near discontinuities. They are based on a special interpolation technique that, when applied to piecewise smooth solution, provides high-order accuracy. ENO strategies are widely used for problems that develop shocks, such as convection-dominated problems, turbulent flow, and shock interaction with vortices. Further details are provided in [21].
- d) **Weighted Essentially Non-Oscillatory methods (WENO)** The first WENO scheme was developed by Liu, Chan, and Osher in 1994 [69], in which a third-order accurate finite volume WENO scheme was designed. Instead of using just one stencil as in the original ENO, WENO methods use a convex combination of all stencils. Reconstructed values at the interfaces are evaluated by merging polynomials and weights, which are calculated using a specific equation. These weights depend on the smoothness of the solution [90]. Both ENO and WENO have been successfully used to simulate shock turbulence interactions [105, 106, 107].

2.4. Mesh Generation and Adaptation

2.4.1. Mesh Generation

Mesh generation is a fundamental step in Computational Fluid Dynamics (CFD), serving as the process of dividing the computational domain into discrete elements or cells. This division, known as a mesh, provides the framework on which the governing equations, such as the Navier-Stokes equations, are discretized and solved numerically. The mesh plays a crucial role in determining the resolution and accuracy of the simulation, as well as its computational efficiency. High-quality mesh generation is, therefore, essential for obtaining reliable and accurate results in CFD simulations [114].

The process of mesh generation begins with defining the geometry of the domain and subdividing it into smaller, simpler elements such as triangles, quadrilaterals, tetrahedra, or hexahedra. These elements serve as the building blocks of the mesh and are selected based on the complexity of the geometry and the desired resolution. Several algorithms are used in this process, including Delaunay triangulation, advancing front techniques, and octree-based methods. Delaunay triangulation, for example, ensures that no mesh element contains points within its circumscribing circle, leading to well-shaped elements that minimize numerical errors. Advancing front methods generate meshes by sequentially constructing elements from the boundaries inward, making them suitable for irregular geometries.

Meshes are broadly classified into three main types: structured, unstructured, and multiblock meshes. Structured meshes are characterized by their regular connectivity, where elements are arranged in a grid-like pattern with rows and columns. In two-dimensional structured meshes, the elements are typically quadrilaterals, while three-dimensional structured meshes are composed of hexahedral elements [118]. Their regular structure allows for efficient computation and straightforward implementation of numerical schemes, making them suitable for domains with simple geometries. However, their regularity imposes limitations when adapting to complex geometries. Unstructured meshes, on the other hand, have no fixed connectivity pattern, enabling them to conform to highly irregular geometries with localized refinements. These meshes often use triangular elements in two dimensions and tetrahedral elements in three dimensions, though other element shapes such as prisms or pyra-

mids may also be employed [54]. The flexibility of unstructured meshes makes them indispensable for simulations involving intricate geometries, but their irregularity leads to increased computational cost and memory requirements compared to structured meshes. Multiblock meshes combine the advantages of both structured and unstructured meshes by dividing the domain into multiple blocks, each of which may independently adopt a structured or unstructured connectivity. This hybrid approach provides a balance between computational efficiency and adaptability, making multiblock meshes well-suited for simulations involving regions with varying geometric complexity [115].

Mesh quality significantly impacts the accuracy and efficiency of CFD simulations. Metrics such as aspect ratio, orthogonality, skewness, and smoothness are used to evaluate the quality of a mesh. Poor-quality meshes can lead to numerical errors, slower convergence rates, or even simulation failure. High-quality meshes ensure better resolution and stability, enabling precise simulation results. For example, a mesh with low skewness and high orthogonality is less likely to introduce errors in the solution, particularly in regions with steep gradients. Advancements in mesh generation tools have greatly enhanced the process, enabling the creation of high-quality meshes with minimal manual intervention. Modern software programs, such as ANSYS Meshing, Gmsh, and OpenFOAM, provide automated tools for generating structured, unstructured, and hybrid meshes tailored to specific applications [114]. Adaptive meshing techniques further enhance accuracy by dynamically refining the mesh in regions where higher resolution is needed during the simulation [74]. For instance, in simulations of turbulent flow, adaptive meshing can refine regions near boundaries to better capture the effects of boundary layers.

Mesh visualization plays an essential role in identifying errors or areas requiring refinement. Modern visualization tools enable engineers to inspect the distribution, quality, and resolution of the mesh before running simulations. This process helps detect potential issues, such as poorly shaped elements or inadequate resolution in critical regions, ensuring better results in the subsequent analysis.

Real-world applications highlight the importance of choosing the right mesh type. For example, structured meshes are commonly used in aerodynamic simulations of aircraft wings, where the geometry is relatively simple, and high computational efficiency is required. Unstructured meshes are often applied in biomedical simulations, such

as modeling blood flow in arteries, where the complex geometry necessitates high adaptability. Multiblock meshes find applications in turbomachinery, where different regions of the domain, such as blades and casings, require varying resolutions and computational strategies [9].

Despite these advancements, challenges in mesh generation persist, particularly in handling highly complex geometries or balancing the trade-off between computational cost and solution accuracy. Emerging technologies, such as machine learning and artificial intelligence, offer promising solutions to optimize the mesh generation process, potentially reducing computational effort while improving mesh quality.

In conclusion, mesh generation is a critical component of CFD, forming the foundation for accurate and efficient simulations. From structured grids for simple geometries to unstructured meshes for intricate domains, and hybrid approaches for flexibility, the choice of mesh depends on the specific requirements of the problem. As technology evolves, the tools and techniques for mesh generation continue to advance, expanding the possibilities for increasingly sophisticated CFD applications.

2.4.2. Mesh Adaptation

Mesh adaptation involves refining the computational mesh to improve solution accuracy. The most common types of mesh refinement are h -refinement, which reduces the mesh size by increasing the number of elements, r -refinement, which redistributes nodes to active solution regions and p -refinement, which locally increases the polynomial order of the approximation within elements. A combination of h - and p -refinements, referred to as hp -refinement, offers enhanced adaptability [62].

The refinement process relies on indicators or sensors to identify zones requiring refinement, with *a posteriori* error estimates being a common choice. Research on robust *a posteriori* error estimators for partial differential equations has been conducted by Roy [96], Houston and Süli [44], and Mitchell and McClain [76].

The selection of a suitable refinement indicator depends on factors such as computational cost, required accuracy, and robustness, particularly for adaptive algorithms in turbulent flow simulations [77]. Refinement indicators can generally be classified into three categories. Feature-based indicators are derived from physical or theoretical properties of the problem and are inexpensive and simple to implement, but they

lack robustness, may not account for error propagation, and require expert users for optimal results. Discretization-Error based indicators identify refinement zones using high error values, estimated as the difference between numerical and exact solutions. However, this approach often leads to suboptimal refinement, as error transport can cause refinement in regions where errors accumulate rather than originate. Goal-oriented indicators focus on reducing the numerical error in a specific target functional, offering the lowest computational cost to achieve required accuracy when the target functional is the simulation goal.

Naddei et al. [77] evaluated the performance of various refinement indicators for p -adaptation simulations of steady and unsteady flows using discontinuous Galerkin methods. Their study analyzed the impact of different indicators on convergence, computational efficiency, and refinement region selection.

2.5. Conclusion

In this chapter, we provided an overview of Computational Fluid Dynamics (CFD), outlining its fundamental principles and methodologies. We began with an introduction to CFD as a powerful computational tool for simulating and analyzing fluid flow in diverse applications. The discussion then focused on the Navier-Stokes equations, which serve as the mathematical foundation for modeling fluid dynamics, emphasizing their importance in capturing the behavior of fluids under various conditions.

We also introduced several numerical discretization methods. Low-order methods, such as finite difference and finite volume methods, were discussed for their straightforward implementation and computational efficiency. High-order methods, such as the Discontinuous Galerkin method, were highlighted for their ability to achieve greater accuracy and effectively handle more complex flow phenomena. Finally, we examined the role of mesh generation and adaptation in CFD simulations, emphasizing the importance of constructing grids that balance accuracy and computational cost. Adaptive mesh techniques were highlighted as a means of dynamically refining grids to improve solution quality in critical regions.

This comprehensive overview of CFD principles and techniques provides a solid foundation for understanding the integration of advanced machine learning approaches to enhance simulation accuracy and efficiency, which will be explored in the following chapters.

Chapter 3

Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on building systems capable of learning from data and improving their performance without explicit programming. This learning process can be categorized into different types, such as supervised, unsupervised, semi-supervised, and reinforcement learning. Each type of learning addresses specific problems and is associated with unique methodologies.

In supervised learning, the algorithm is trained on labeled data, enabling it to make predictions about unseen data. Common algorithms in this category include support vector machines, decision trees, and neural networks. Unsupervised learning, on the other hand, deals with unlabeled data, where the goal is to uncover hidden patterns or structures in the data. The Gaussian Mixture Model (GMM), k-means clustering, and principal component analysis (PCA) are examples of unsupervised learning methods. Semi-supervised learning combines elements of both supervised and unsupervised learning, while reinforcement learning involves training an agent to make sequential decisions by interacting with an environment.

This chapter introduces the fundamental concepts of machine learning, explores different types of learning in [Section 3.1](#). [Section 3.2](#) highlights key algorithms, particularly those relevant to unsupervised methods such as GMM, which will be applied later to enhance CFD.

3.1. Learning Types in Machine Learning

Machine learning classifiers fall into four primary categories based on the kind of learning: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning.

3.1.1. Supervised Learning

This type of learning is conducted under the supervision of a teacher. Labeled data sets are used to train algorithms, that classify data or predict results (outputs) accurately. In this learning process, the input vector is presented to the network, which will give an output vector. This output vector is compared with the desired output vector, then the model adjusts its weights until the actual output matches the desired output [100]. In 3.1, a sketch of the supervised learning is shown.

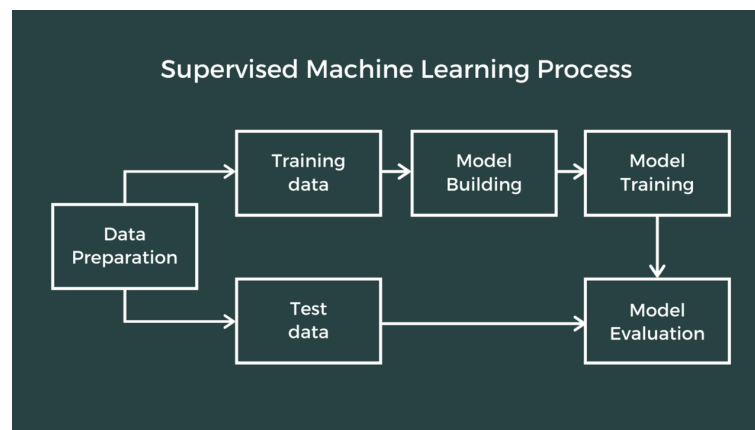


Figure 3.1.: Supervised Learning process, taken from [1]

3.1.2. Unsupervised Learning

As the name suggests, unsupervised machine learning uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns, create labels or data groupings without the need for human intervention. It provides the springboard for conducting further analysis after discrete groups have been discovered.

Its ability to detect unclassified categories by discovering similarities and differences

in information makes it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition [113].

Unsupervised learning models are generally used in clustering tasks, association, and dimensionality reduction tasks.

It uses a specific objective function (e.g., a function that minimizes distances within a class to find tight groups). K-methods and hierarchical clustering are among the most widely used unsupervised clustering techniques [24].

3.1.3. Semi-supervised Learning

Semi-supervised learning is also known as classification with labeled and unlabeled data (or partially labeled data). It is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data.

The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. This type of machine learning improves the supervised learning tasks using readily available unlabeled data in the case that labeled data is scarce, absent or expensive. Semi-supervised learning can also be used as a quantitative tool for understanding human category learning, where most of the inputs are self-evidently unlabeled.

Some popular semi-supervised learning models, includes self-training, mixture models, co-training and multi-view learning, graph-based methods, and other semi-supervised approaches [129].

3.1.4. Reinforcement Learning

Reinforcement learning (RL) is the training of machine learning models to make a sequence of decisions. It differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected, after choosing an action. The agent is told the immediate reward and the subsequent state but is not told which action would have been in its best

long-term interests. Here the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge) [113].

The agent in RL is the component that makes the decision of what action to take. For instance, for a robot, the agent is typically not the whole robot, but the specific program running on the robot's CPU that makes the decision on the action. All the relays/motors and other parts of the physical body of the robot are parts of the environment in RL terms, the objective of the RL is that the agent learns to achieve a goal in an uncertain, potentially complex environment.

In reinforcement learning, artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Reinforcement machine learning is concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward, its goal being to maximize the total reward [45].

In Figure 3.2, a sketch of a RL process is shown. On each step of interaction, the agent receives as input “i” and some indication of the current state of the environment “s”. The agent then chooses an action “a” to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal ‘r’. The agent’s behavior “B” should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms. There are no absolute restrictions, but if the reward function is “better behaved”, the agent will learn better. Practically, this means speed of convergence, and not getting stuck in local minima. But further specifications will depend strongly on the species of reinforcement learning that are being used, for example, is the state/action space continuous or discrete? Is the world or the action selection stochastic? Is reward continuously harvested, or only at the end?

Reward policy is set without giving the model any suggestion for how to do its task. It begins with the development of random trials until it reaches a more advanced and efficient performance by maximizing the reward.

Taking advantage of the power of search and many trials, the agent gathers useful

experience about the possible system states, actions, transitions and rewards actively to act optimally [52].

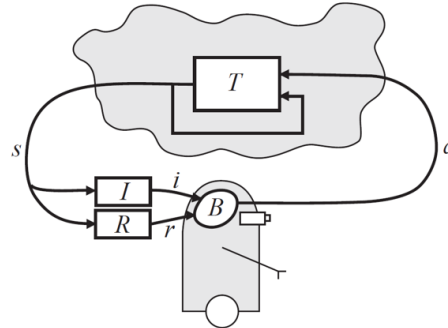


Figure 3.2.: The standard reinforcement-learning model, taken from [52]

3.2. Machine Learning Algorithms

Machine Learning tasks are guided by a wide variety of algorithms that organize the learning process. These algorithms are broadly categorized into supervised and unsupervised learning approaches. Supervised learning includes algorithms such as Linear Regression, Support Vector Machines, and Neural Networks. Unsupervised learning focuses on finding patterns and structures in data without labeled outputs and includes algorithms like K-means clustering and Gaussian Mixture Models (GMM).

3.2.1. Supervised Learning Algorithms

Linear Regression Regression is a way to model target value based on independent predictors. Regression analysis is a common technique used to analyze multifactorial data. It uses an equation to express the relationship between a variable of interest (response) and a set of related predictor variables [112].

Support Vector Machines The aim of the support vector machine algorithm is to find a hyperplane in N -dimensional space that precisely classifies the data points. There are many possible hyperplanes that could be chosen to separate two classes of data points [37].

Neural Networks Neural Networks (NN) or artificial neural networks (ANN) are formed by artificial neurons or nodes to solve artificial intelligence problems. They can be trained through a dataset, or learning can occur within networks, which is called Self-learning. NNs can derive conclusions from a complex and seemingly unrelated set of information. Artificial Neural Network system is borrowed from the analogy of biological neural networks. ANNs are also named as ‘artificial neural systems’, or parallel distributed processing systems’, or connectionist systems’. ANN acquires a large collection of units (neurons or nodes) that operate in parallel. These units are interconnected by links, and each connection link is associated with a weight that has information about the output signal, the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units [2].

A formal neuron (see 3.3) is a mathematical function; it is conceived as a model of biological neuron. Formal neurons are elementary units in an artificial neural network.

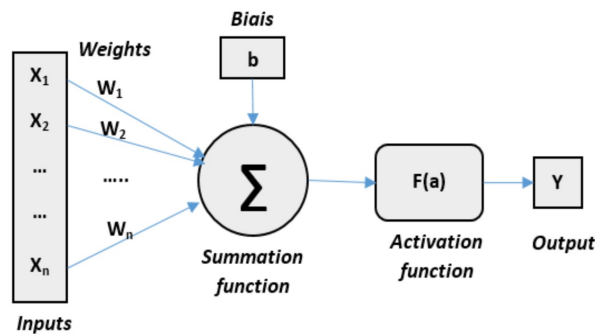


Figure 3.3.: Mathematical model of the formal neuron, taken from [101]

Where the output

$$a = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \dots + x_m \cdot w_m + b.$$

This output can be calculated by applying the activation function over the net input.

$$y = F(a),$$

Neural Networks can be classified as the following types: Feedback ANN (or recurrent

or interactive where the information travel in both directions), Feed Forward ANN (the information travel in one way from input to output). There are many architectures of ANNs such as convolutional Neural Networks, Recurrent Neural Networks and Autoencoders:

- Convolutional Neural Networks: (ConvNets or CNNs) are one of the various types of neural nets. CNNs are more often used for classification and computer vision tasks. Convolutional neural networks provide a more scalable approach to image classification and object recognition tasks, using principles from linear algebra, to identify patterns within an image. CNNs are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, Convolutional layer which can be followed by a Pooling layer then the Fully-connected (FC) layer as a final layer.

Most of the computation occurs in the convolutional layer, that is why it is considered as the most important layer in CNNs. This process requires input data, a kernel (filter or feature detector which is a two-dimensional (2-D) array of weights) and a feature map or activation map or convolved feature which is the output. A dot product is fed into an output array after it has been calculated between the filter (typically a 3x3 matrix) and a local area of the input image (receptive field). This process is repeated while the filter shifts by a stride, until the kernel has swept across the entire image.

ConvNet aims to reduce the size of the images without losing features which are critical for getting a good prediction, to facilitate the data processing. At the end of convolution operation, a Rectified Linear Unit ($ReLU = \max(0, x)$, where x is a real number) transformation is applied to the feature map, introducing nonlinearity to the model.

As in convolutional layers, pooling layers (downsampling or conducts dimensionality reduction) are also used for reducing the number of parameters in the input. Similarly, a filter moves across the input image applying an aggregation function to the values within the receptive field instead of the dot product used earlier in the convolutional layer. This aggregation function can be either a max function (max pooling) or an average function (average pooling) which are the two main types of pooling. Even though a lot of information is lost during

the pooling process, it helps to reduce complexity, improve efficiency and limit overfitting risks.

The last type of layer introduced here is the fully-connected layer. From its name one can expect that each node in the output layer is directly connected to another node in the previous layer. Unlike convolutional and pooling layers, where the information in the input image are partially connected to the output layer, this layer is effective in classification tasks using the extracted patterns from the previous layers. Typically a softmax activation function that returns the probability of each input to be classified into binary or multi-classes is used. This activation function is defined as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)},$$

where z is the result of a dot product between the input and a set of weights that pass through the softmax activation function to provide the final output, more details about CNNs are in [2]. In Figure 3.4 is described the CNNs architecture.

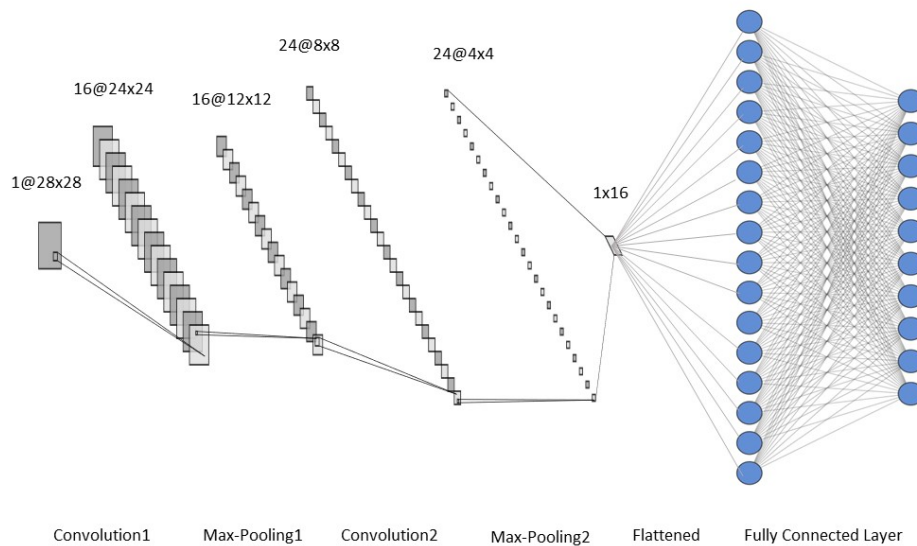


Figure 3.4.: Convolutional Neural Network architecture (CNNs)

- **Recurrent Neural Networks:** RNNs, described in Figure 3.5, are different types of ANNs, usually adopted by voice search, Google translate and some other

applications to carry out translation and natural language processing (NLP), speech recognition and other tasks. They use time-series data and sequential data for the training process. RNN is characterized by its ability to store previous information, it considers the current input and also what it has learned from the inputs it received previously to predict new output. This is what makes it much better than other ANNs types in solving temporal problems [79].

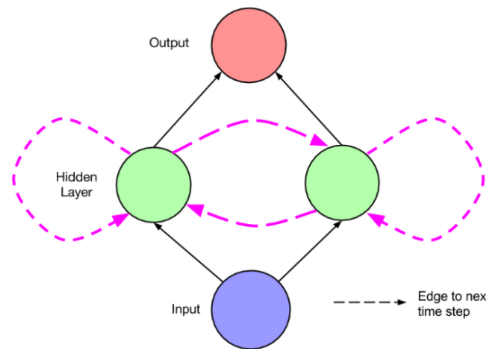


Figure 3.5.: Recurrent Neural Network, taken from [66]

- **Autoencoders:** These are models used in unsupervised learning to represent data in a smaller dimension by discovering its underlying correlations, these models are trained the way of supervised learning problems. An autoencoder as shown in Figure 3.6 has three main parts: an encoder, where the input is compressed to create the lower-dimensional hidden layer, a bottleneck that contains the most interesting complex relationships in data, and then the model pass on the third part ‘the decoder’, which reconstructs the input. To do that a loss function (named Reconstruction Loss) has to be minimized, it is presented by the error between the reconstructed output and the input [2].

3.2.2. Unsupervised Learning Algorithms

Unsupervised learning encompasses a range of techniques designed to uncover patterns and structures in data without the need for labeled examples. Unlike supervised learning, where the algorithm learns from input-output pairs, unsupervised learning algorithms attempt to find inherent groupings, associations, or representations in the dataset. Clustering, in particular, is one of the most widely used unsupervised learning techniques, with applications in regression, prediction, data mining, biology,

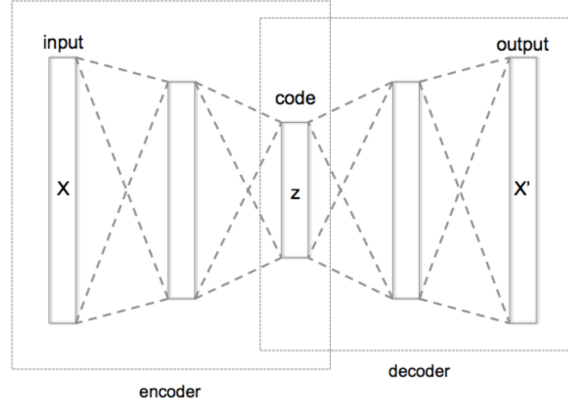


Figure 3.6.: Autoencoder, taken from [40]

genetics, and manufacturing.

A well-known definition of clustering is provided by Webster [122], who describes it as a statistical classification technique used to determine whether individuals in a population fall into distinct groups based on quantitative comparisons of multiple characteristics. Several studies, such as those by Rokach [95] and Jain [49], have explored different clustering approaches and their applications in diverse fields. Clustering techniques are often categorized into hard and soft clustering approaches. Hard clustering methods, such as K-means, assign each data point to a single cluster, while soft clustering methods, such as Gaussian Mixture Models (GMM), assign probabilities to each point belonging to different clusters. Various distance metrics, such as Euclidean distance, are used to measure similarity between data points, which play a fundamental role in the clustering process.

Mathematically, each data point is represented as a feature vector in an n -dimensional space. Given two patterns, $\mathbf{x}$ and $\mathbf{y}$, the Euclidean distance between them is calculated as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (3.1)$$

This distance measure is fundamental in clustering algorithms that rely on centroid-based or density-based approaches.

Several popular clustering techniques include:

K-means Clustering K-means is a simple and widely used clustering algorithm that partitions the data into k clusters. It iteratively assigns each data point to the nearest cluster center and updates the cluster centers as the mean of the assigned points. The process is repeated until convergence, often using a loss function such as the sum of squared distances between points and their corresponding cluster center:

$$J = \sum_{i=1}^k \sum_{j \in C_i} \|x_j - \mu_i\|^2,$$

where C_i is the set of points assigned to cluster i , and μ_i is the mean of those points [72].

Gaussian Mixture Models (GMM) GMM is a probabilistic model that represents the data as a mixture of multiple Gaussian distributions. Each Gaussian component is defined by a mean vector and a covariance matrix, and the model assigns probabilities to each data point belonging to different components. The parameters of the GMM are learned using the Expectation-Maximization (EM) algorithm, which iteratively maximizes the likelihood function of the observed data:

$$p(x) = \sum_{i=1}^k \pi_i \mathcal{N}(x | \mu_i, \Sigma_i),$$

where π_i is the weight of the i -th Gaussian component, μ_i is its mean, and Σ_i is its covariance matrix [7]. GMM is particularly useful in applications where the data exhibits multimodal distributions, such as image segmentation and anomaly detection.

Fuzzy Clustering Fuzzy clustering introduces a nuanced approach, allowing patterns to belong to multiple clusters with varying degrees of membership. The fuzzification process can be represented by membership values:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}}}$$

where μ_{ij} represents the membership of data point j in cluster i , d_{ij} is the distance between data point j and the centroid of cluster i , and m is a fuzzification parameter. Fuzzy clustering provides a more nuanced view of membership, allowing for soft boundaries between

clusters.

Cluster Evaluation and Applications A critical step in clustering involves determining the optimal number of clusters. Techniques like the *elbow method*, which examines the rate of change of variance explained by increasing clusters, and *silhouette analysis*, which measures how similar an object is to its own cluster compared to other clusters, help users in making informed decisions [46].

Both K-means and GMM are widely used clustering methods. K-means is computationally efficient and works well for convex-shaped clusters, whereas GMM provides more flexibility by allowing for soft clustering and capturing complex, multimodal distributions. These methods are widely applied in diverse domains, including image processing, bioinformatics, finance, and engineering, where identifying meaningful patterns in large datasets is crucial.

3.3. Conclusion

This chapter introduced the fundamental principles of machine learning, establishing a basis for understanding its application in solving complex computational problems. Section 3.1 explored the different types of learning, including supervised, unsupervised, and reinforcement learning, highlighting their distinctions and use cases. These categories were discussed to provide a clear understanding of how machine learning methods are structured and applied based on the problem context. Section 3.2 reviewed key machine learning algorithms, with a particular focus on unsupervised learning techniques such as *K-means clustering* and *Gaussian Mixture Models (GMM)*. These clustering methods play a crucial role in identifying patterns within unlabeled data, making them particularly useful for applications in *data analysis and adaptive computational strategies*. The discussion also highlighted the strengths and limitations of different clustering approaches, comparing their effectiveness in various scenarios.

By integrating these fundamental concepts, this chapter sets the stage for the application of machine learning in *enhancing CFD simulations*. The techniques introduced here, particularly clustering-based methods, will be used in subsequent chapters to enable *adaptive mesh refinement and efficient computational resource allocation*. The insights gained from this discussion provide a foundation for understanding how

data-driven strategies can complement numerical methods in solving complex fluid dynamics problems.

Part II.

Enhancing CFD by Machine Learning

Summary:

In this part, we explore the integration of machine learning techniques into computational fluid dynamics (CFD) simulations. Through our investigation, we aim to enhance the efficiency and accuracy of CFD simulations using the capabilities of machine learning algorithms. Specifically, we focus on the implementation of a Clustering-based P-Adaptation Method, as well as the utilization of Shock-Capturing-based Machine Learning approaches.

Chapter 4

Computational and Data-Driven Frameworks

This chapter explores the tools and techniques that form the foundation of the methodology employed in this study. It introduces the computational and data-driven frameworks used to address challenges in Computational Fluid Dynamics (CFD) simulations. Section 4.1 provides an overview of the Discontinuous Galerkin Spectral Element Method (DGSEM), which is implemented in the HORSES3D solver and offers a high-fidelity computational framework for simulating complex fluid-flow phenomena. In Section 4.2, the chapter then discusses Gaussian Mixture Models (GMM), a probabilistic clustering approach optimized using the Expectation-Maximization (EM) algorithm. GMM plays a vital role in analyzing CFD data, pinpointing key flow regions, and enabling adaptive mesh refinement to enhance computational efficiency.

By focusing on the computational framework and clustering techniques, this chapter lays the groundwork for the methodology detailed in the next chapters. It explains how these tools are employed to enhance simulation accuracy and efficiency, offering insights into their respective roles and integration into the overall workflow.

4.1. Computational Fluid Dynamics

4.1.1. High-order discontinuous Galerkin solver

The data used in this work have been generated using the high-order spectral element CFD solver HORSES3D [29]. HORSES3D is a 3D parallel code developed at ETSIAE-UPM (the School of Aeronautics of the Polytechnic University of Madrid). This framework uses the high-order discontinuous Galerkin spectral element method (DGSEM) and is written in modern Fortran 2003. It targets simulations of fluid flow phenomena, such as those governed by compressible and incompressible Navier-Stokes equations, and it supports curvilinear hexahedral meshes of arbitrary order. There are several options of subgrid turbulence models for the users to choose from for LES simulations such as the Smagorinsky subgrid model, WALE and Vreman. One of the main features offered in HORSES3D is its support for anisotropic p non-conforming elements, as the polynomial order can be varied independently in each element in each direction. In this work, we use the adaptation capability to increase/decrease the polynomial order uniformly within each element in the regions determined by the clustering technique. All cases are run with low storage Runge-Kutta 3 time advance [124].

4.1.2. Discontinuous Galerkin Spectral Element Method (DGSEM)

The Discontinuous Galerkin Spectral Element Method (DGSEM) is a high-order numerical method used to solve partial differential equations (PDEs), particularly in Computational Fluid Dynamics (CFD). DGSEM is a hybrid approach that combines the advantages of the Discontinuous Galerkin (DG) method and the Spectral Element Method (SEM), enabling high-fidelity simulations of complex fluid flows with improved computational efficiency. Rooted in the DG method, DGSEM retains key principles such as element decomposition, discontinuous representation at element boundaries, and local conservation. However, it distinguishes itself through the introduction of spectral basis functions, often orthogonal polynomials like Legendre or Chebyshev, within each element. This combination enhances the method's accuracy and enabling it to achieve high-order precision efficiently. DGSEM is well-suited for problems with shocks, complex geometries, where traditional methods may struggle. However, DGSEM still retains the advantageous exponential convergence property

that is characteristic of spectral element methods [30, 99].

The computational domain is discretized into non-overlapping curvilinear hexahedral elements, each mapped onto a reference element using a polynomial transfinite mapping [58]. This transformation ensures that numerical computations are carried out in a standard computational domain. The transformation of the governing equations from physical space to the reference space is given by:

$$J \frac{\partial u}{\partial t} + \nabla_{\xi} \cdot F_e = \nabla_{\xi} \cdot F_v \quad (4.1)$$

where J represents the Jacobian of the mapping from physical to reference space, ∇_{ξ} is the differential operator in the reference space ξ , F_e corresponds to inviscid fluxes, and F_v represents viscous fluxes. The variable u denotes the vector of conservative variables, such as density, momentum components, and total energy.

The solution is approximated within each element using spectral basis functions, typically Legendre or Gauss-Lobatto polynomials. The solution representation in each element is expressed as:

$$u_h(x, t) = \sum_{k=1}^N U_k(t) \Phi_k(x), \quad (4.2)$$

where $u_h(\xi, t)$ is the approximate numerical solution, $U_k(t)$ are the nodal values at the interpolation points ξ_k , and $\Phi_k(\xi)$ are the Lagrange basis functions defined at the Legendre–Gauss–Lobatto points in the reference element.

The index N denotes the polynomial degree of the approximation.

While Equation 4.2 represents the approximation in one spatial dimension, in practice, DGSEM is applied in multiple dimensions by forming tensor products of one-dimensional basis functions along each coordinate direction.

The weak formulation of DGSEM is obtained by multiplying the governing equation by a test function ϕ_j and integrating over the reference element:

$$\int_{\text{element}} J u_t \phi_j d\Omega + \int_{\text{element}} \nabla_{\xi} \cdot F_e \phi_j d\Omega = \int_{\text{element}} \nabla_{\xi} \cdot F_v \phi_j d\Omega. \quad (4.3)$$

where ϕ_j denotes the test function with $0 < j < N$, and $d\Omega$ is the differential volume element in the reference domain.

Applying integration by parts to the divergence terms, shifting the derivative onto the test function and introducing the boundary terms:

$$\int_{\text{element}} J u \phi_j d\Omega - \int_{\text{element}} \nabla_{\xi} \cdot F_e \phi_j d\Omega = \int_{\partial \text{element}} F_e^* \cdot \hat{n} \phi_j dS - \int_{\text{element}} \nabla_{\xi} \cdot F_v \phi_j d\Omega. \quad (4.4)$$

where $\hat{n}$ denotes the outward-pointing unit normal vector to the boundary $\partial \text{element}$, and dS is the differential surface element on the element boundary.

In practice, the viscous term is not integrated by parts, as its proper discretization in DG methods requires additional treatment. Techniques such as the Bassi–Rebay methods (BR1 or BR2) or the Interior Penalty (IP) approach are commonly employed to ensure stability and consistency. To avoid these complexities, we retain the simple volume form of the viscous fluxes, as done in the HORSES3D solver [30].

Since DG allows discontinuities at element boundaries, the inviscid flux F_e at the interface is not uniquely defined. Instead, a numerical flux F_e^* is introduced, computed using a Riemann solver to ensure conservation:

$$\int_{\partial \text{element}} F_e^* \cdot \hat{n} \phi_j dS. \quad (4.5)$$

where F_e^* is the approximate numerical flux at the interface. This flux is computed using schemes such as Roe flux solver (for inviscid interactions), Bassi–Rebay (BR1) method (for viscous interactions). These fluxes are chosen because of their stability and conservation properties in high-order schemes [30].

To enhance computational efficiency, DGSEM can be used in a hybrid approach where different equations are solved in different regions of the domain. The governing equations are selected based on a clustering technique, leading to a division of the computational domain into viscous/Turbulent Regions where the full Navier–Stokes equations are solved and inviscid Regions in which the Euler equations are solved, reducing computational cost [83]. This classification is performed using a machine

learning-based clustering method, specifically the Gaussian Mixture Model (GMM). By identifying key flow regions, this technique allows for a more adaptive simulation strategy [83].

Additionally, DGSEM supports *P-adaptation*, where the polynomial order of the solution representation is dynamically adjusted in different regions. This ensures high accuracy in critical areas while optimizing computational resources[99].

DGSEM is well-suited for parallel computing due to its element-wise discontinuous nature. The implementation in HORSES3D utilizes a low-storage Runge-Kutta scheme for time integration, reducing memory requirements while maintaining accuracy [58]. The incorporation of clustering-based decomposition and adaptive resolution has demonstrated significant computational savings (41% reduction in computational cost for airfoil simulations [38]. and 45% reduction in computational cost for wind turbine simulations).

This efficiency makes DGSEM a powerful tool for high-fidelity simulations in CFD, particularly for applications requiring accurate resolution of boundary layers, wakes, and turbulent structures.

The discontinuous element representation and spectral accuracy, it offers superior resolution for complex flow phenomena. The integration of machine learning-based domain decomposition and adaptive polynomial refinement enhances computational efficiency while maintaining high-fidelity results. This makes DGSEM particularly effective for aerodynamic simulations, where adaptive resolution is crucial to capture turbulent and shock-induced flow features.

4.2. Machine Learning

4.2.1. Data Analysis and Clustering with Gaussian Mixture Models: A Python Framework

This subsection outlines a comprehensive Python framework for clustering data using Gaussian Mixture Models (GMM). The framework encompasses the libraries, methodologies, and steps required to use GMM for our data clustering.

Libraries

- **Pandas:** Used for data manipulation and analysis, providing data structures like DataFrame.
- **Matplotlib:** Primary Python plotting library to create static, interactive, and animated visualizations.
- **NumPy:** Fundamental package for scientific computing in Python, facilitating array manipulation and mathematical operations.
- **Seaborn:** Built on top of Matplotlib, Seaborn offers enhanced visualization capabilities and aesthetic appeal.
- **SciPy:** Scientific computing library that extends NumPy with additional functionalities, including optimization and statistical algorithms.
- **Scikit-learn:** Comprehensive Python machine learning library, providing tools for data preprocessing, modeling, and evaluation. Specifically, the 'Gaussian-Mixture' module from scikit-learn is utilized to implement GMM.

Methodology of the Algorithm

The core methodology of this framework is based on Gaussian Mixture Models (GMM) for data clustering. GMM is a probabilistic model particularly useful when clusters exhibit overlapping distributions or when distinct boundaries between clusters are not well defined.

Gaussian Mixture Model (GMM): The foundation of the algorithm is the assumption that the data points originate from a mixture of multiple Gaussian distributions. Each Gaussian component represents a cluster, and the model estimates the parameters, mean, covariance, and mixture coefficients, to best fit the data. Unlike traditional clustering techniques, GMM enables soft clustering by assigning each data point a probability of belonging to different clusters, allowing for a more nuanced understanding of the data structure.

Expectation-Maximization (EM) Algorithm: The EM algorithm is the primary optimization technique used to estimate the parameters of the Gaussian components. It operates iteratively through two main steps:

- **Expectation (E) Step:** Calculate the probability that each data point belongs to each cluster based on the current parameter estimates.
- **Maximization (M) Step:** Updates the parameters, mean, covariance, and mixture coefficients—based on the computed cluster assignments.

This process continues until convergence is reached, ensuring that the model parameters stabilize and the likelihood of the data under the model is maximized.

Clustering and Analysis: Once the GMM model is trained, it is used to assign cluster labels to data points based on the highest probability of belonging to a cluster. In addition, cluster probabilities provide insight into the confidence levels of these assignments, offering valuable information about potential overlaps between clusters.

Framework Implementation

The framework follows a structured sequence of steps to preprocess the data, perform clustering, and analyze the results.

The process begins with **data loading and preprocessing**, where necessary libraries are imported, and the dataset is loaded into a Pandas DataFrame. At this stage, missing values are handled, data types are converted, and column names are standardized to ensure a clean dataset. Next, **feature extraction and transformation** are performed, where relevant features are selected and transformed to enhance their suitability for clustering. To ensure uniform scale among features, **data scaling** is applied using techniques such as StandardScaler from scikit-learn. With the data preprocessed, the **Gaussian Mixture Model (GMM)** is initialized and fitted using the GaussianMixture module from scikit-learn. The number of clusters (components) and convergence criteria are defined to optimize the clustering process. After training, the model predicts **cluster labels** for each data point, which are then assigned to the original dataset. Additional insights, such as cluster probabilities, are calculated to evaluate the confidence of each assignment. Finally, **post-processing and visualization** techniques are applied to interpret the clustering results effectively. Unique identifiers can be extracted and cluster statistics calculated to gain deeper insight. Visualization tools such as Matplotlib and Seaborn are employed to create plots that illustrate the clustered data, facilitating interpretation and further analysis.

Building on this, the proposed framework provides a structured approach to data

clustering using Gaussian Mixture Model (GMM) in Python, offering a robust method for analyzing complex datasets and identifying underlying patterns.

4.2.2. Gaussian Mixture Model

Gaussian Mixture Models (GMM) are a probabilistic approach to clustering that assumes data points are generated from a mixture of multiple Gaussian distributions. Unlike hard clustering methods like k-means, GMM allows for soft clustering, meaning each data point is assigned a probability of belonging to each cluster rather than being exclusively assigned to a single group. This flexibility makes GMM particularly useful for datasets where clusters have overlapping regions or are not clearly separated. Each Gaussian component in GMM is defined by its mean, covariance, and a mixture weight that represents its contribution to the overall distribution. The number of Gaussian components, typically determined using methods like the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC), defines the number of clusters in the dataset [103].

To estimate the parameters of the Gaussian components (means, covariances, and mixture weights), GMM employs the Expectation-Maximization (EM) algorithm, an iterative optimization technique that alternates between assigning probabilistic cluster memberships and updating model parameters to maximize the likelihood of the data.

4.2.2.1. GMM with EM Algorithm

One of the robust framework for clustering datasets characterized by intricate underlying structures is the Gaussian Mixture Model (GMM) with the Expectation-Maximization algorithm forms[94, 117].

The Gaussian Mixture Model (GMM) constitutes a pivotal framework in statistical data analysis, presuming that the observed dataset arises from a composite distribution of Gaussian components. Through a clustering methodology, GMM discerns discernible sub-populations within the dataset, each adhering to its distinctive Gaussian distribution. The probability density function which is a weighted sum of a finite mixture of normal distributions, encapsulates the inherent data structure by synthesizing diverse Gaussian components, providing an intricate portrayal of complex data distributions.

Simultaneously, the Expectation-Maximization (EM) algorithm, rooted in likelihood-based inference, addresses parameter estimation in models featuring incomplete or latent data. Its foundational principles involve maximizing the expected complete data likelihood through iterative Expectation and Maximization steps. The algorithm, applicable to various statistical models, exhibits monotonic increases in observed data likelihood, ensuring convergence to a local maximum. The versatility and theoretical underpinnings of the EM algorithm render it a valuable tool for handling incomplete data in diverse statistical and machine learning applications. The GMM clustering procedure entails parameter estimation, involving means, covariances, and weights affiliated with each Gaussian component. This iterative process provides a comprehensive understanding of the underlying substructures within the dataset. To understand how the algorithm identifies the optimal set of parameters to characterize the mixture.

4.2.2.2. Probability Model for Observing Data

The probability of observing any data point, x_i , in a GMM is given by a weighted sum of K Gaussian distributions:

$$P(x_i|\Theta) = \sum_{j=1}^K \alpha_j \mathcal{N}(x_i|\mu_j, \Sigma_j)$$

Where:

- $\Theta = \{\alpha_j, \mu_j, \Sigma_j\}_{j=1}^K$ represents the parameters of the GMM.
- α_j is the mixture weight for the j -th Gaussian component.
- μ_j is the mean of the j -th Gaussian component.
- Σ_j is the covariance matrix of the j -th Gaussian component.
- $\mathcal{N}(x|\mu, \Sigma)$ is the Gaussian distribution with mean μ and covariance matrix Σ .

4.2.2.3. Log-Likelihood and Log-Likelihood Maximization

The log-likelihood of the dataset, given the parameters Θ , is the natural logarithm of the joint probability:

$$\log P(X|\Theta) = \sum_{i=1}^N \log P(x_i|\Theta)$$

The goal is to maximize this log-likelihood, or equivalently, the log-likelihood per data point:

$$\frac{1}{N} \log P(X|\Theta)$$

4.2.2.4. Expectation-Maximization (EM) Algorithm Steps

The Expectation-Maximization (EM) algorithm is a powerful iterative optimization technique widely used in statistical modeling and machine learning. The primary idea behind EM is to iteratively refine estimates of model parameters in the presence of latent or unobservable variables. In the context of Gaussian Mixture Models (GMMs), the EM algorithm is employed to iteratively improve the parameter estimates of the model.

The EM algorithm consists of two main steps: the Expectation Step and the Maximization Step. These steps alternate until the algorithm converges to a stable set of parameter values that maximize the likelihood of the observed data. The convergence ensures that the model is capturing the underlying structure of the dataset effectively.

Expectation Step

In the Expectation Step, the algorithm computes the responsibility of each data point for each cluster. The responsibility (r_{ij}) signifies the probability that data point x_i belongs to cluster j . This step involves assessing the likelihood of each data point belonging to each cluster based on the current parameters of the GMM.

1. Responsibility for Cluster Assignment:

$$r_{ij} = \frac{\alpha_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}{\sum_{k=1}^K \alpha_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}$$

Here, α_j is the mixture weight for the j -th Gaussian component, and $\mathcal{N}(x_i|\mu_j, \Sigma_j)$ is the probability density function of the Gaussian distribution with mean μ_j and covariance matrix Σ_j .

Maximization Step

The Maximization Step involves updating the parameters of the GMM based on the responsibilities computed in the Expectation Step. This step maximizes the expected log-likelihood with respect to the parameters.

2. Update Mixture Weights (α_j):

$$\hat{\alpha}_j = \frac{1}{N} \sum_{i=1}^N r_{ij}$$

3. Update Means (μ_j):

$$\hat{\mu}_j = \frac{\sum_{i=1}^N r_{ij} x_i}{\sum_{i=1}^N r_{ij}}$$

4. Update Covariance Matrices (Σ_j):

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^N r_{ij} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\top}{\sum_{i=1}^N r_{ij}}$$

In these equations, N is the total number of data points, and r_{ij} represents the responsibility computed in the Expectation Step.

The EM algorithm iterates between these two steps, updating the responsibilities and refining the model parameters, until a convergence criterion is met. It provides an effective way to estimate the parameters of complex models, especially when dealing with incomplete or latent data.

4.2.2.5. Jensen's Inequality and Lower Bound

Jensen's Inequality is a fundamental concept used in the EM algorithm to establish a lower bound on the log-likelihood. In the context of GMMs and EM, this inequality is applied to the expected log-likelihood, where $q(t)$ is the posterior distribution over latent variables.

$$\log E[q(t)] \geq E[\log q(t)]$$

This inequality ensures that each iteration of the EM algorithm results in an increase in the lower bound on the log-likelihood, contributing to the overall convergence of the algorithm.

4.2.2.6. Constraints in Optimization

Optimizing GMMs involves adhering to certain constraints on the model parameters to ensure the validity of the solution. These constraints are particularly crucial for the mixture weights (α_j) and covariance matrices (Σ_j).

- **Mixture Weights Constraints:** Mixture weights must be non-negative and sum to 1. This ensures that each cluster's contribution is appropriately weighted in the overall model.
- **Covariance Matrices Constraints:** Covariance matrices must satisfy conditions for being positive definite. This constraint guarantees that the covariance matrices are well-behaved and reflect valid relationships between features within each cluster.

Adhering to these constraints is essential for obtaining meaningful and interpretable results from the GMM optimization process.

4.3. Conclusion

In this chapter, we have outlined the fundamental tools and techniques that underpin the methodology of this study. The high-order Discontinuous Galerkin Spectral Element Method (DGSEM), as implemented in the HORSES3D solver, was presented as a robust computational framework for high-fidelity simulations of complex fluid-flow phenomena. Additionally, Gaussian Mixture Model (GMM) were introduced as a powerful data-driven approach for clustering and analyzing CFD data, enabling adaptive mesh refinement and resource optimization.

By establishing the capabilities and applications of these tools, this chapter sets the stage for the methodological developments described in the following chapters. Together, these frameworks illustrate the synergy between advanced numerical methods and machine learning, forming the foundation for addressing the challenges in CFD simulations explored throughout this work.

Chapter 5

Feature detection using the Gaussian Mixture Model

The accurate identification of distinct flow regions in computational fluid dynamics (CFD) is essential for advancing numerical techniques and improving simulation outcomes. This chapter focuses on the evaluation of feature detection methods used to separate flow regions, serving as a foundation for their subsequent application in mesh adaptation, which will be discussed in the next chapter.

Section [5.1](#) presents key flow sensors frequently employed in academic research for flow region separation. These include traditional methods such as the Q-criterion and vorticity-based sensors. Additionally, a newly proposed clustering method, based on the Gaussian Mixture Model (GMM), is explored. This method facilitates the automatic separation of the flow field into two distinct regions: viscous/rotational (encompassing viscous and turbulent flows) and inviscid/irrotational (representing potential flow), without requiring predefined thresholds. The section concludes with a description of the test cases used to validate these methods. Section [5.2](#), titled Evaluation and Application of Detection Methods, focuses on a comprehensive evaluation of the identified regions. It compares the performance of the GMM-based methodology with traditional approaches, analyzing the impact of user-defined parameters and assessing the quality of the detected regions.

By thoroughly examining these feature detection techniques, this chapter establishes a framework for their application in mesh adaptation strategies, contributing to the overarching goals of this research.

5.1. Methodology and Tools for Feature Detection

This section presents some flow sensors frequently employed for flow region separation in academic research, including Q-criterion and vorticity-based sensors. Additionally, we will explore a newly proposed clustering method deemed advantageous for flow regions detection. This method allows for the separation of the flow field into two distinct regions: viscous/rotational (comprising viscous+turbulent flows) and inviscid/irrotational (representing potential flow), thereby eliminating the necessity for predefined thresholds. Subsequently, we conduct a comprehensive evaluation of regions detected using our methodology and the two traditional methodologies.

5.1.1. Feature-based sensors for region separation

5.1.1.1. Q-criterion sensor:

The Q-criterion, first introduced in [47], This sensor relies on the rotational and strain rate tensors and can be expressed in a non-dimensional form as illustrated in [53]:

$$f = \frac{1}{2} \left(\frac{\|\boldsymbol{\Omega}\|^2}{\|\mathbf{S}\|^2} - 1 \right). \quad (5.1)$$

Here, $\mathbf{S}$ and $\boldsymbol{\Omega}$ denote the strain and rotational rate tensors, respectively, calculated as $\mathbf{S} = (\mathbf{J} + \mathbf{J}^T)/2$ and $\boldsymbol{\Omega} = (\mathbf{J} - \mathbf{J}^T)/2$, where $\mathbf{J} = \nabla \mathbf{U}$ represents the gradient of the velocity field $\mathbf{U} = (u, v, w)$.

Additionally, it is noteworthy that the Q-criterion has been effectively employed in prior research endeavors, as exemplified in [53], particularly for the aim of mesh adaptation.

The sensor f is utilized to detect rotations within the flow field. Specifically, when $f \rightarrow -\frac{1}{2}$, the Q criterion identifies regions where the strain rate exceeds the vorticity in the flow, whereas the values of $f \gg -\frac{1}{2}$ signify the dominance of rotation.

The Q-criterion effectively partitions the flow into two distinct regions by establishing a specific demarcation based on a threshold value. In this scheme, the values of $f < K$ indicate an irrotational region, while the values of $f > K$ denote a rotational region.

Moreover, the non-dimensional form of the Q-criterion offers the advantage of mini-

mizing variability in Q-criterion values across different test cases. For example, in [53], the authors used a refinement threshold of $K = 1$ in their simulations. This threshold is tailored to be universally applicable across a wide range of flow fields, and requires no user intervention.

However, even a slight adjustment in the threshold value K can lead to significant changes in the detected regions, as will be shown in Section 5.2.1.

5.1.1.2. New Ω sensor:

The second considered sensor is the 'new Ω method' introduced in [67], which is a specialized technique for detecting vortices in wake flows. This sensor considers the ratio of the norm of the vorticity tensor to the sum of the norms of the vorticity tensor and the strain rate tensor using the following equation:

$$g = \frac{||\mathbf{\Omega}||^2}{||\mathbf{\Omega}||^2 + ||\mathbf{S}||^2 + \epsilon}, \quad \epsilon = 10^{-3}. \quad (5.2)$$

In this framework, as $g \rightarrow 0$, it denotes an irrotational region, while $g \gg 0$ indicates regions of high rotation. Utilizing this sensor facilitates the determination of rotation-dominated regions requiring mesh refinement [86]. The identification of elements within the domain necessitating refinement is achieved by adjusting a threshold $K \in [0, 1]$ based on the function $g > K$ or $g < K$. Moreover, the 'new Ω method' aims to minimize user intervention in defining the threshold value. In [67], a constant value of $K = 0.52$ is utilized across different test cases. Subsequently, in [86], a slightly different refinement threshold of $K = 0.7$ is employed. Nevertheless, in both instances, it is asserted that minimal adjustment of the threshold is necessary between simulations.

However, the efficacy of this identification method relies on the intensity of the vortical structures within the flow, as will be illustrated in section 5.2.2.

5.1.1.3. Gaussian mixture model for flow region detection:

Finally, we investigate the innovative clustering technique using the Gaussian Mixture Model (GMM) as introduced in [84]. This approach utilizes the principal invariants of strain and rotational rate tensors as input. The methodology, explained in Subsection 4.2.2, is designed to differentiate between two specific flow regions: one

characterized by viscous rotational flow and the other by inviscid rotational flow. The utilization of this method for both mesh adaptation and shock capturing has not been explored previously, and we will explore these aspects in the upcoming sections.

The feature space utilized as input to the GMM relies on Galilean invariants of the flow, denoted as $E = \{Q_s, R_s, Q_\Omega\}$, where Q_s and R_s represent the two principal invariants of the strain rate tensor $\mathbf{S}$, defined as:

$$Q_s = \frac{1}{2} \left((\text{tr}(\mathbf{S}))^2 - \text{tr}(\mathbf{S}^2) \right), \quad (5.3)$$

$$R_s = -\frac{1}{3} \det(\mathbf{S}). \quad (5.4)$$

The primary invariant Q_s identifies regions where there is a high local viscous dissipation rate of the kinetic energy $\epsilon = -4\mu Q_s$ [127], where μ represents the dynamic viscosity of the fluid flow. Conversely, R_s represents the second invariant of the strain rate tensor, indicating strain production or destruction [22]. The rotational rate tensor $\mathbf{\Omega}$ is defined by a single invariant:

$$Q_\Omega = -\frac{1}{2} \left(\text{tr}(\mathbf{\Omega}^2) \right). \quad (5.5)$$

Q_Ω helps identify rotational regions within the flow field, as it is directly related to enstrophy density ξ [127].

The GMM produces two variables, p_v and p_i , indicating the likelihood of each point in the mesh belonging to a viscous rotational or inviscid irrotational region. It is crucial to highlight that the feature space utilized in this approach comprises variables that are distinctive for regions dominated by viscous rotational flow. Consequently, this technique functions on the premise that the feature space variables exhibit higher values in regions with viscous rotational flow regions and lower values in inviscid irrotational flow regions. A point is categorized as viscous if $p_v \geq p_i$, and as inviscid otherwise. More details can be found in [84].

In contrast to prior sensors, this technique does not require any pre-defined threshold. Therefore, the accuracy of this classification is not affected by the intensity of vortices in the flow, as will be elaborated in Section 5.2.

For the particular case of 2D flows, we still consider 3×3 matrices ($\mathbf{J}$, $\mathbf{S}$ and $\mathbf{\Omega}$), but these

contain zeros in the third row and column (since the spanwise flow and associated gradients are zero). Therefore, 3D and 2D matrices encapsulate the same physical information, resulting in $R_s = 0$ and the feature space $E = \{Q_s, R_s = 0, Q_\Omega\}$, which is used to train the GMM clustering algorithm for 2D flows.

The GMM along with the feature space E is used to detect two flow regions, a rotational viscous-dominated region and an inviscid irrotational region. We note that it is possible to include in our feature space an additional invariant $\text{tr}(\mathbf{S})$ associated with continuity/compressibility effects. However, in this chapter, we only consider incompressible flows (Mach numbers < 0.2) and therefore this invariant does not provide relevant information and is omitted.

5.1.2. Description of test cases

The test cases considered in this study are utilized in Section 5.2.1 to compare our methodology with other traditional approaches documented in the literature. They are also employed in Chapter 6 to assess the efficacy of this methodology for mesh adaptation. Our analysis of the influence of the threshold value on region definition for traditional sensors is applied to three scenarios: a steady laminar flow test case around a cylinder at $Re = 40$, an unsteady laminar flow around a cylinder at $Re = 100$, and an unsteady turbulent flow around a cylinder at $Re = 3900$ as described in Section 5.1.2.

- **Reynolds 40:** We begin by examining a steady laminar flow past a circular cylinder at $Re = 40$, this test case has been thoroughly studied in previous researches [25, 34, 18]. These findings will be used as a reference point for evaluating our results. The data of this test case is generated using a grid comprising 684 hexahedral components with 85,500 degrees of freedom.
- **Reynolds 100:** The second test case we considered to investigate our study is an unsteady laminar flow around a circular cylinder at $Re = 100$, the flow in this test case is unsteady with a shedding frequency of $T=6U_c/D$ non-dimensional time units. This flow regime has been extensively examined by several researchers in the context of computational simulations [68, 10, 15, 91, 98] and serves as a

basis for comparison. The mesh utilized consists of 1282 hexahedral elements.

- **Reynolds 3900:** This particular case has been thoroughly examined by various researchers in both numerical [87, 71] and experimental domains [87, 80, 70]. In this scenario, a second-order mesh comprising 20,736 hexahedral elements is used. The mesh has been extended in the spanwise direction with $L_z/D = \pi$ and subdivided into 16 elements along this direction. To minimize aliasing errors and ensure robustness, a split-form discretization method has been employed along with the Pirozzoli averaging [92] and a Smagorinsky closure model. These methodologies collectively contribute to enhancing the precision and robustness of the approach.

5.2. Evaluation and Application of Detection Methods

5.2.1. Analysis of the effect of the user-defined threshold

In this part of the study, we perform a preliminary assessment of the three sensors that were introduced earlier. We focus on robustness with respect to the user-defined threshold. We compare the Q-criterion and the new Ω method with the clustering technique to demonstrate how the new methodology outperforms the classical sensors for the three test cases considered, without the need to use a user-defined and problem-dependent threshold.

In the study, the three previously introduced test cases were simulated until the wake is fully developed. All simulations were run with a grid resolution of $P = 4$. Subsequently, these data were utilized to compute the sensors (Q-criterion, the new Ω method and the GMM cluster), and the results were juxtaposed.

For unsteady scenarios, a single snapshot of the fully developed flow was utilized, consistent with the methodology outlined in [84]. It is important to mention that multiple snapshots were evaluated, revealing no significant differences in the clustered regions.

- **Q-criterion:** We conducted the identification of flow regions using multiple thresholds denoted as K for the Q-criterion in three distinct test cases: $Re = 40$, $Re = 100$, and $Re = 3900$. The results are presented in Figures 5.1, 5.2, and 5.3.

Additionally, we provide the regions identified by the clustering method, which utilizes GMM with the feature space denoted as E in Figures 5.1d, 5.2d, and 5.3d.

Initially, by setting the threshold at $K = 1$, as recommended in [53], only a few elements are selected for the second cluster across our test cases, as depicted in Figures 5.1c, 5.2c, and 5.3c. Subsequently, to ensure a comparable number of selected elements for cluster 2 as with the clustering technique, adjustments were made to the threshold values for the Q-criterion. Specifically, for $Re = 40$, we set $K = -0.453$; for $Re = 100$, $K = -0.193$; and for $Re = 3900$, $K = -0.4679$. This modification resulted in an increased number of identified elements for the second cluster, as illustrated in Figures 5.1a, 5.2a, and 5.3a. However, there is a significant difference in the region marked when using the Q-criterion with these thresholds compared to using the clustering technique.

In the aim of automating the threshold selection process for the Q-criterion, we employed the GMM algorithm utilizing the Q-criterion as the feature space. This resulted in threshold values of $K = -0.4909$ for $Re = 40$, $K = 0.343$ for $Re = 100$, and $K = 0.4014$ for $Re = 3900$. The identified elements marked for cluster 2 through this automated process are depicted in Figures 5.1b, 5.2b, and 5.3b.

For $Re = 100$ and $Re = 3900$, where a higher density of vortices is observed, our automated approach yielded threshold values closer to those documented in the literature. However, at $Re = 40$, where the flow remains steady, the automatic threshold selection by GMM differs significantly from the literature. This indicates that the literature threshold value struggles to accurately discern regions in cases of steady flow.

The quality of the meshes generated with Q-criterion for different thresholds will be analyzed later in Sections 5.2.2 and 6.2.1

It can be seen that the detected regions for each cluster change considerably when we change the Q-criterion thresholds, which confirms the sensitivity to the thresholds. Moreover, the Q-criterion misidentifies far-field cells for adaptation when considering $K = 0.453$ and $K = 0.4909$ for the automatic threshold computed with GMM. Also, note that the most reasonable thresholds are different for each Reynolds number: for the flow at $Re = 40$ is $K \approx -0.2$,

$K \approx 0.2$ for the flow at $Re = 100$, and $K \approx 0$ for the flow at $Re = 3900$. In contrast, the GMM clustering and the feature space E is an unsupervised process that does not involve a selection of a threshold value and can automatically select cells near the wall and the wake region, where the flow field contains considerable viscous and rotational effects, as shown in figures 5.1d, 5.2d and 5.3d.

- **New Ω method:** We conducted the flow region identification using the second sensor considered new Ω . Similar to the Q-criterion, different thresholds are tested for the new Ω method across the three test cases. Figures 5.4, 5.5, and 5.6 show the results of the region identification.

Furthermore, for comparison, we present the regions identified by the GMM clustering technique with the feature space E , as illustrated in Figures 5.4d, 5.5d, and 5.6d.

Initially, the selection of thresholds for the new Ω method was automated using the GMM algorithm. This process yielded values of $K = 0.0156$ for $Re = 40$, $K = 0.0013$ for $Re = 100$, and $K = 0.0003$ for $Re = 3900$. The marked elements for refinement resulting from this automated procedure are presented in Figures 5.4b, 5.5b, and 5.6b.

Following this, we adjusted the threshold values for the new Ω method to ensure a comparable number of selected elements for refinement as with the clustering technique. Specifically, for $Re = 40$, K was set to 0.1; for $Re = 100$, $K = 0.04625$; and for $Re = 3900$, $K = 0.03278$. This adjustment led to an increased number of elements marked for the second cluster correspondent to high values of the function f defining the sensor, as illustrated in Figures 5.4a, 5.5a, and 5.6a. It is notable that in this case as well, the identified regions differ between the new Ω method with these thresholds and the clustering technique.

In comparison, initially, a threshold of $K = 0.7$ was set for the new Ω method, as suggested in [86]. However, this resulted in the selection of very few elements for the cluster with high values of the sensor across our test cases, as observed in Figures 5.4c, 5.5c, and 5.6c.

The disparities in threshold selection by the GMM and the literature default were

most pronounced at $Re = 40$, where the flow remains steady. This deviation persisted at $Re = 100$, where the automated threshold also included more elements for the second cluster with high values of the sensor than the literature default. This discrepancy was attributed to difficulties in distinguishing between elements in the far field and wake, as their new Ω levels exhibited similarities. This behavior was particularly noticeable at $Re = 3900$, resembling the $Re = 100$ case, where a large number of far-field elements were included in the cluster with high values of the new Ω sensor due to these identification challenges.

In contrast, our proposed GMM clustering, utilizing the feature space E , represents an unsupervised method that eliminates the necessity of manually setting a threshold value. It autonomously separates cells proximal to the wall and wake region, crucial areas encompassing viscous/turbulent effects within the flow.

This study highlights the significant variability observed in the detected regions when adjusting the thresholds of traditional feature-based sensors, such as the Q-criterion and the new Ω method.

In the next chapter 6, will conduct an analysis to perform mesh adaptation in the regions identified by the clustering technique, comparing them with those identified by traditional feature-based sensors.

5.2.2. Assessment of the quality of the identified regions

In the previous section, the feasibility of achieving comparable refinement regions through the three methods under comparison was demonstrated. The marked regions were initially derived using GMM with feature space E . Following this, adjustments were made to the thresholds including the thresholds that give an identical number of elements to those obtained through the clustering technique for both the Q-criterion and the new Ω sensors to mark flow regions. However, despite these efforts, distinct regions were detected for the clustering technique, the Q-criterion, and the new Ω method.

This section embarks on a comparative analysis of these regions, aimed at assessing their suitability for mesh adaptation. Through this comparison, a comprehensive understanding of the disparities in the identified regions between traditional feature-

based sensors and the new clustering technique will be elucidated. This examination serves to provide insights into the strengths and weaknesses of each method in accurately identifying regions warranting mesh adaptation. Additionally, it offers valuable implications for the refinement strategies employed in computational fluid dynamics simulations, contributing to advancements in simulation accuracy and computational efficiency.

We present scatter plots depicting the invariants Q_S , R_S , and Q_Ω across the viscous/rotational regions for $Re = 40$, $Re = 100$, and $Re = 3900$. These scatter plots are generated from the same snapshot of the fully developed flow utilized to separate the flow regions. The normalization of the data presented in the plots is based on the maximum data value across the entire domain. This normalization facilitates a clear observation: regions with lower values in the scaled invariants correspond to areas lacking significant viscous/rotational effects.

In our investigation of the cylinder at $Re = 40$, it was determined that the selection of specific values for the parameters K resulted in meshes with a comparable number of degrees of freedom to those obtained through GMM clustering along the feature space $E = (Q_S, R_S, Q_\Omega)$. Specifically, setting $K = -0.0453$ for the Q-criterion and $K = 0.1$ for the new Ω method led to this equivalence. This finding is visually depicted in Figure 5.7, which showcases the scatter map $(Q_\Omega, -Q_S)$ within the identified inviscid region.

For (2D) flows, it is noteworthy that $R_S = 0$, as elaborated upon in [84]. Consequently, the representation of scatter plots for both $(-Q_S, R_S)$ and (Q_Ω, R_S) is deemed unnecessary.

The visualization provided in Figure 5.7 illustrates that the inviscid region, as identified using the Q-criterion and the new Ω method, exhibits heightened levels of viscous dissipation when compared to the region identified through the clustering technique. This observation underscores the limitations of traditional sensors in effectively distinguishing between viscous and inviscid flow regions in contrast to the GMM approach utilizing the feature space E .

In examining the flow past a cylinder at $Re = 100$, the calibration process for the Q-criterion and the new Ω sensors involves setting specific values for the parameters K . Specifically, the thresholds are established as $K = -0.193$ for the Q-criterion and

$K = 0.04625$ for the new Ω method. This calibration procedure ensures that the identified regions with meshes of degree of freedom equivalent to those obtained through the clustering technique.

Figure 5.8 provides visual representation through the scatter map $(Q_\Omega, -Q_S)$ within the inviscid region for both traditional sensors and GMM clustering. This illustration offers insights into the performance of the calibrated sensors in identifying regions with varying levels of viscous dissipation and rotational effects.

The utilization of specific threshold values for the Q-criterion and the new Ω method facilitates the separation of the flow regions that effectively capture the flow dynamics around the cylinder at $Re = 100$. As observed in the $Re = 40$ scenario, there is a discernible increase in viscous dissipation within the inviscid regions identified by traditional sensors relative to those pinpointed by GMM clustering. However, in the case of $Re = 100$, a nuanced difference emerges wherein the new Ω method demonstrates enhanced efficacy compared to the Q-criterion but still lags behind the performance achieved by the clustering technique.

The comparison between the traditional approaches and GMM clustering reveals intriguing insights into the effectiveness of different methodologies in identifying regions with varying levels of viscous dissipation. While the traditional sensors exhibit notable limitations in accurately capturing these regions, the new Ω method showcases advancements in this regard, although not to the extent achieved by the clustering technique.

For the flow past a cylinder at $Re = 3900$, Figure 5.9 shows scatter maps $(Q_\Omega, -Q_S)$, $(-Q_S, R_S)$, and (Q_Ω, R_S) within the inviscid irrotational regions identified using both the Q-criterion and the new Ω sensors, which are fine-tuned with $K = -0.4679$ and $K = 0.03278$, respectively, in comparison to the results obtained through GMM clustering.

The visualizations in Figures 5.9a and 5.9b highlight significantly elevated levels of viscous dissipation within the outer regions delineated by the Q-criterion and the new Ω sensors when contrasted with the clustering technique. Furthermore, an examination of Figures 5.9b and 5.9c reveals that the inviscid irrotational regions identified using traditional sensors exhibit heightened levels of strain production/destruction, with $\frac{R_S}{\max(R_S)}$ values reaching as high as 0.2.

Consequently, these findings suggest that the regions detected by traditional sensors

harbor significant viscous effects compared to those identified by the clustering technique. This underscores the importance of utilizing advanced methodologies, such as GMM clustering, to accurately capture the regions of interest in high-Reynolds-number flow simulations.

The regions identified using the clustering technique, without any threshold adjustments, exhibit superior performance compared to those derived from traditional feature-based methodologies. While it may be feasible to fine-tune thresholds in feature-based approaches to incorporate more elements for refinement and achieve better-suited regions, such adjustments come at the cost of increased computational overhead compared to the clustering methodology.

5.3. Conclusion

In this chapter, we evaluated various feature detection methods for separating flow regions in computational fluid dynamics (CFD) simulations, focusing on their ability to distinguish between viscous/rotational and inviscid/irrotational regions. Traditional methods, such as the Q-criterion and vorticity-based sensors, were assessed alongside a newly proposed clustering method based on the Gaussian Mixture Model (GMM). The GMM-based approach demonstrated the advantage of automating the region separation process without the need for predefined thresholds, offering a robust and flexible alternative to conventional techniques.

The performance of these methods was analyzed using carefully selected test cases, highlighting the strengths and limitations of each approach. A comprehensive evaluation of the identified regions revealed the impact of user-defined thresholds and the overall quality of the detected regions, providing valuable insights into their potential applications.

The findings presented in this chapter form a solid foundation for the application of these feature detection techniques in mesh adaptation, which will be discussed in detail in the following chapter. This integration promises to enhance the efficiency and accuracy of CFD simulations, advancing the broader objectives of this research.

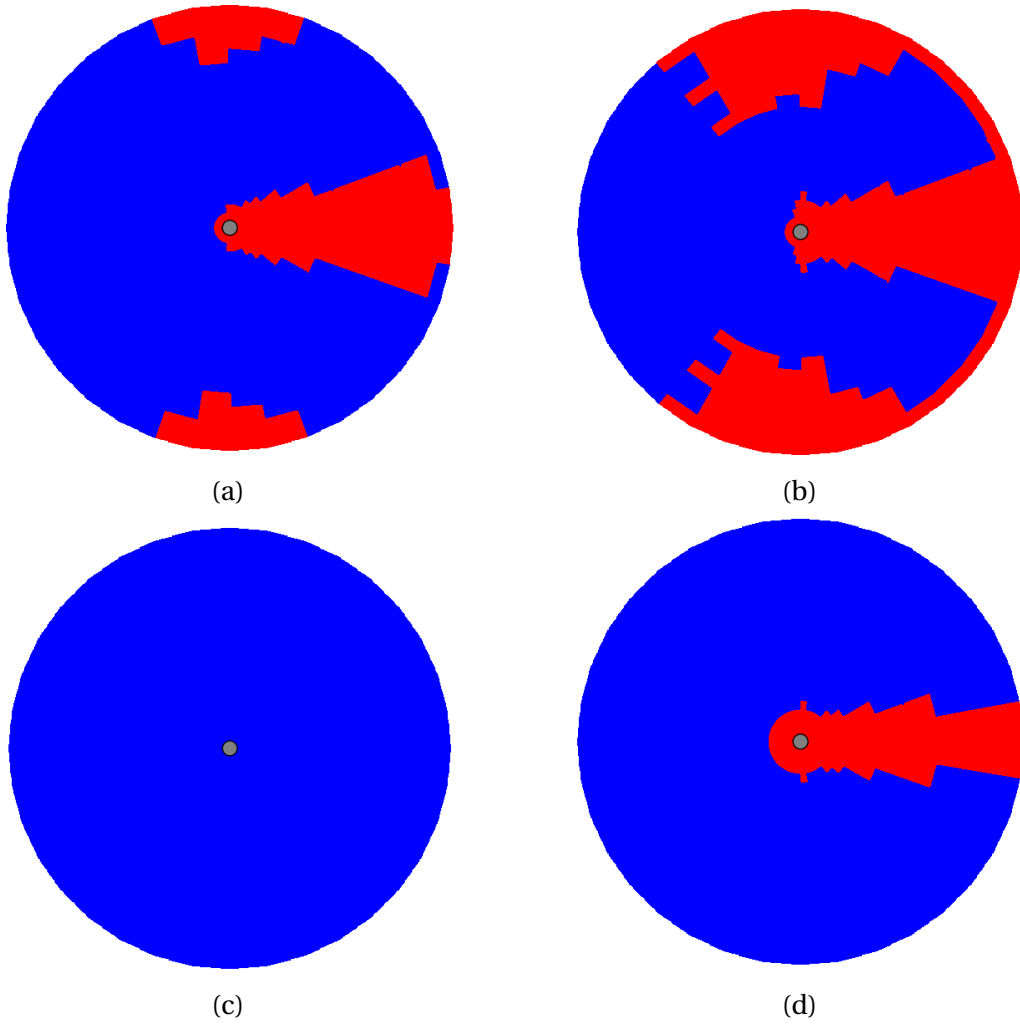


Figure 5.1.: Flow past a cylinder at $Re = 40$: Flow regions selected using the Q-criterion sensor f , $K = -0.453$ (a), $K = -0.4909$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.

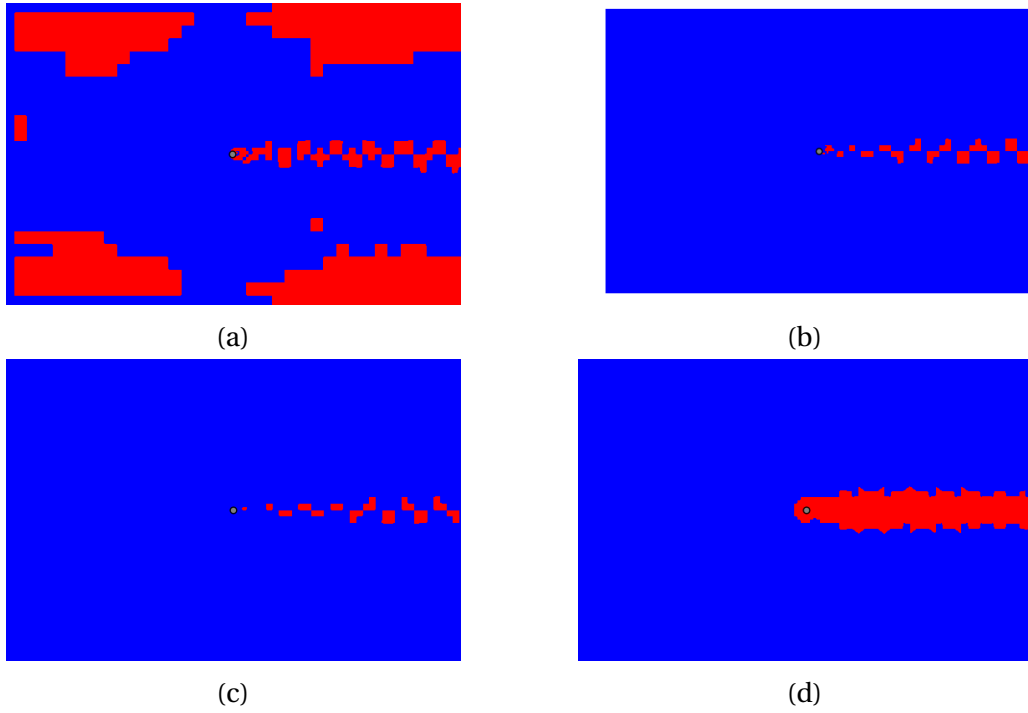


Figure 5.2.: Flow past a cylinder at $Re = 100$: Flow regions selected using the Q-criterion sensor f , $K = -0.193$ (a), $K = 0.343$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.

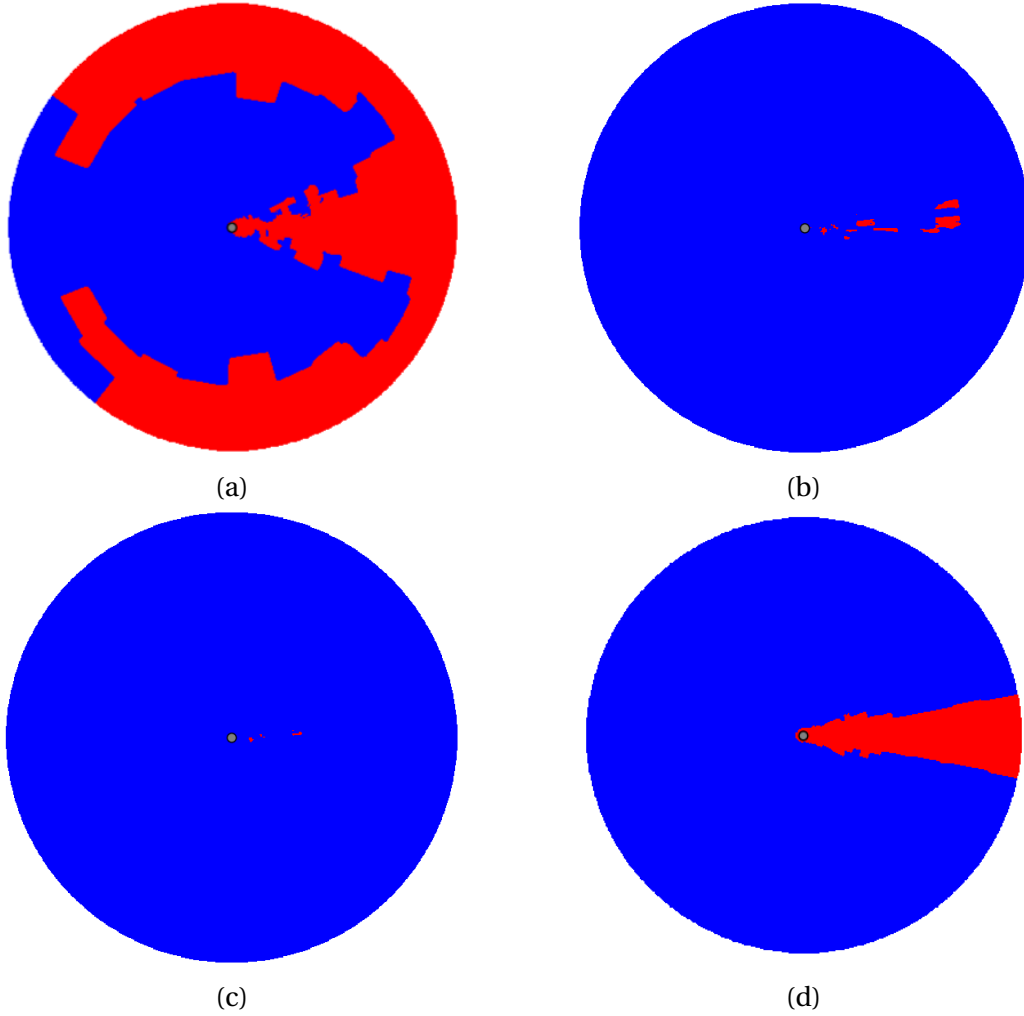


Figure 5.3.: Flow past a cylinder at $Re = 3900$: Flow regions selected using the Q-criterion sensor f , $K = -0.4679$ (a), $K = 0.4014$ - automatic threshold computed with GMM. (b), $K = 1$ - threshold literature default. (c) and GMM clustering with feature space E . (d). **Red**: Regions Viscous rotational region, **Blue**: Inviscid irrotational region.

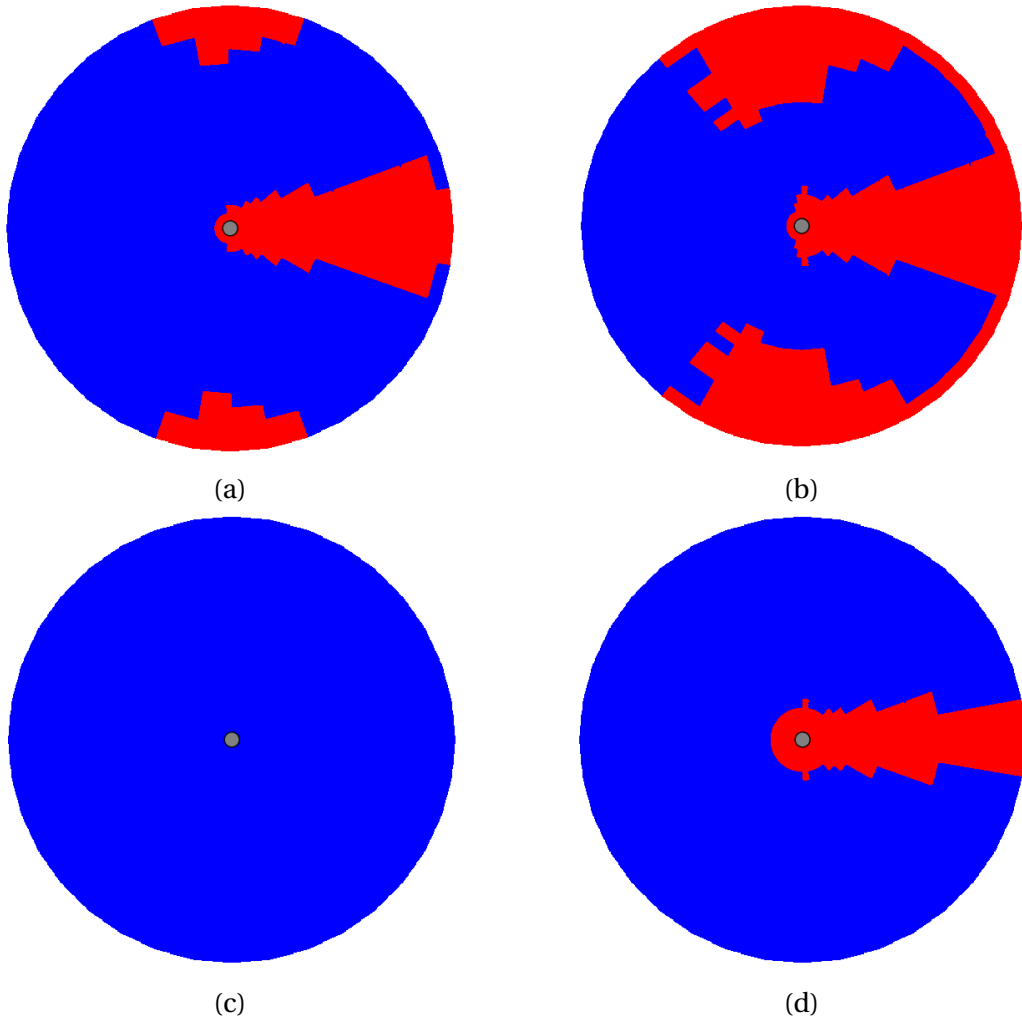


Figure 5.4.: Flow past a cylinder at $Re = 40$: Flow regions selected using the Ω sensor g , $K = 0.1$ (a), $K = 0.0156$ - automatic threshold computed with GMM. (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.

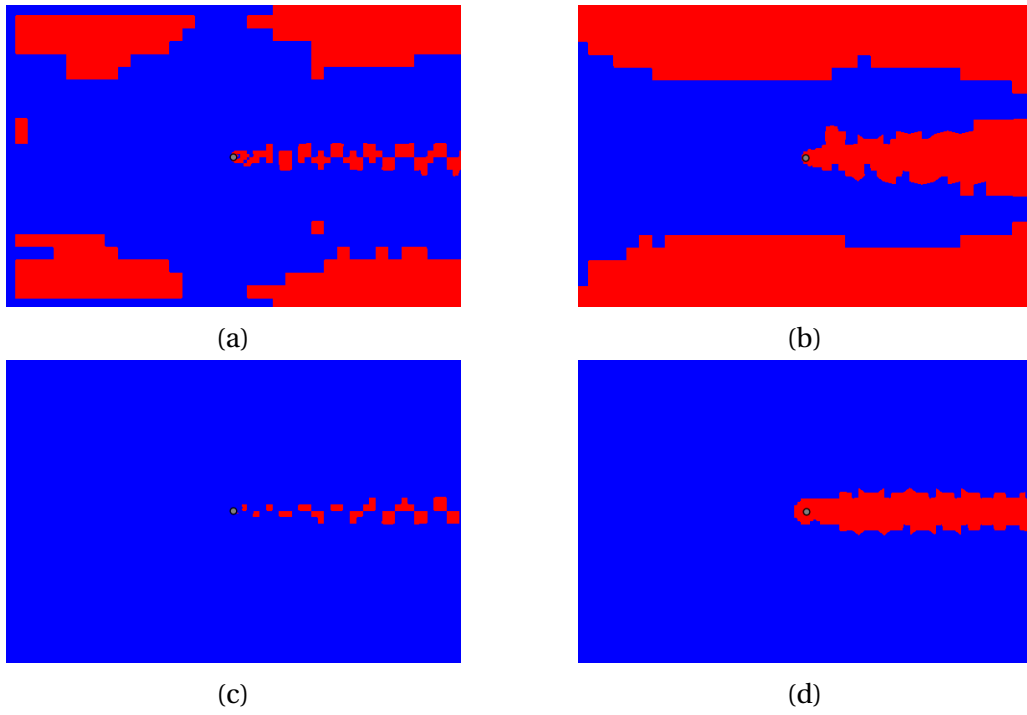


Figure 5.5.: Flow past a cylinder at $Re = 100$: Flow regions selected using the Ω sensor g , $K = 0.04625$ (a), $K = 0.0013$ - automatic threshold computed with GMM. (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E . (d). Red: Viscous region, Blue: Inviscid region.

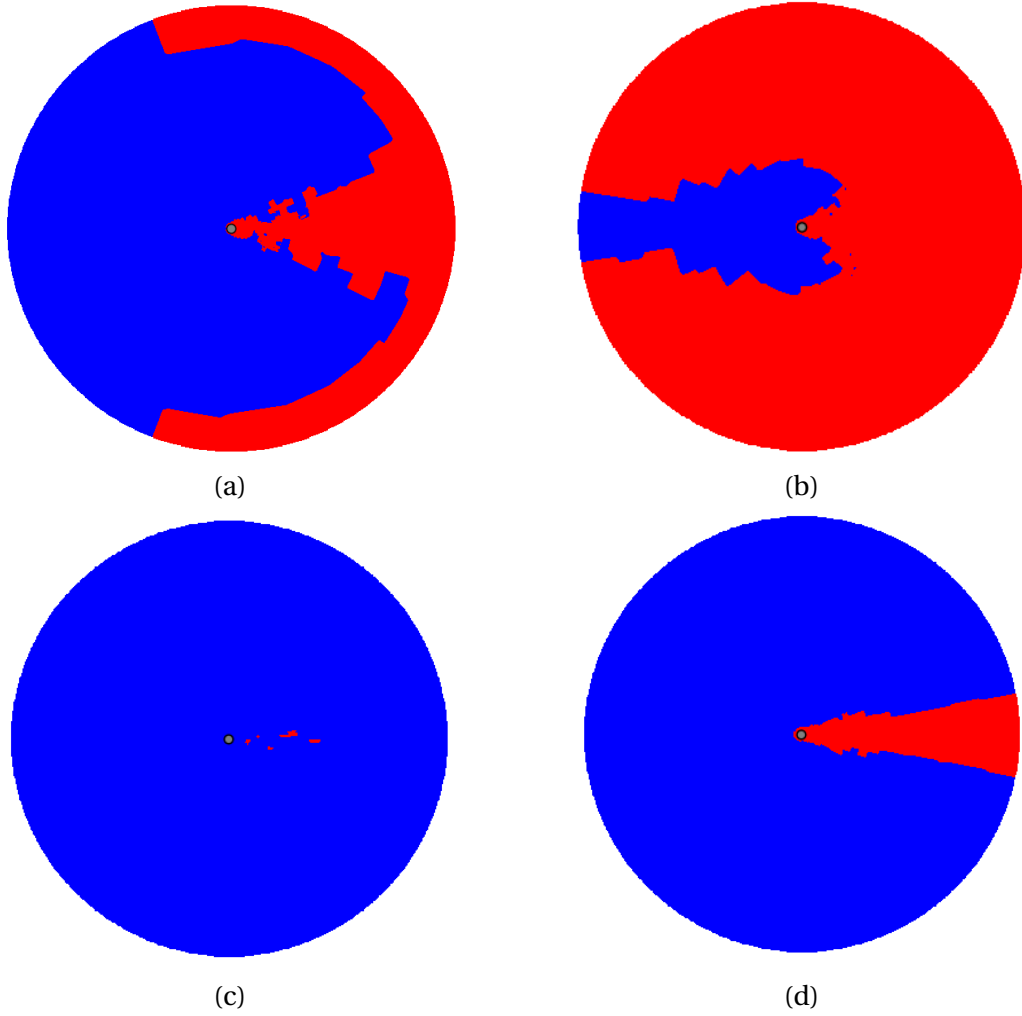


Figure 5.6.: Flow past a cylinder at $Re = 3900$: Flow regions selected using the Ω sensor g , $K = 0.03278$ (a), $K = 0.0003$ - automatic threshold computed with GMM (b), $K = 0.7$ - threshold literature default. (c) and GMM clustering with feature space E (d). Red: Viscous rotational region, Blue: Inviscid irrotational region.

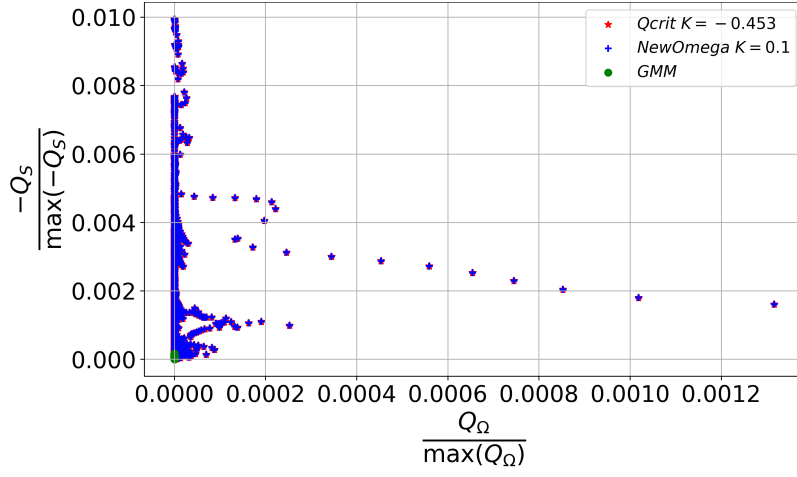


Figure 5.7.: Scatter map $(Q_\Omega, -Q_S)$ in the inviscid region for the flow past a cylinder at $Re = 40$

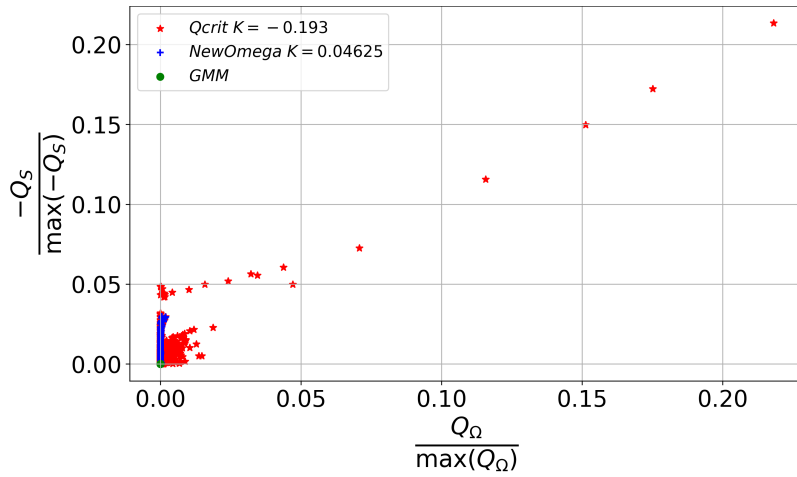


Figure 5.8.: Scatter map $(Q_\Omega, -Q_S)$ in the inviscid region for the flow past a cylinder at $Re = 100$

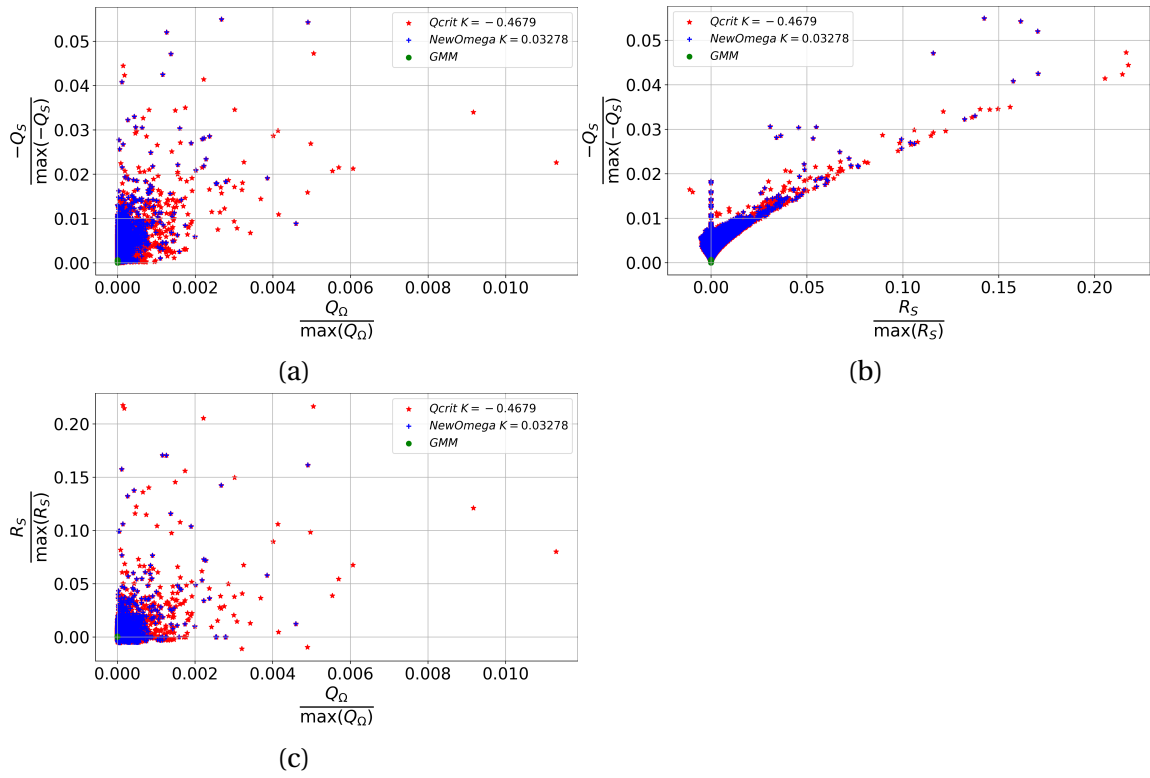


Figure 5.9.: Scatter maps $(Q_\Omega, -Q_S)$, $(R_S, -Q_S)$ and (Q_Ω, R_S) in the inviscid regions for the flow past a cylinder at $Re = 3900$

Chapter 6

A Clustering-based P-Adaptation Method

This chapter explores the application of a Gaussian Mixture Model (GMM)-based methodology to drive p -adaptation in computational fluid dynamics (CFD) simulations. Building on the feature detection techniques presented in earlier chapters, this approach partitions the flow field into two distinct regions—viscous/turbulent and inviscid/irrotational—enabling an efficient and accurate mesh refinement process. The GMM eliminates the need for predefined thresholds, improving adaptability across various flow regimes, and its performance is evaluated through a series of numerical experiments.

The chapter begins with 6.1, which describes the GMM-driven mesh adaptation process. This section explains the clustering techniques employed to partition the flow field and the methodology for assigning polynomial orders based on the detected clusters to optimize computational efficiency while maintaining solution accuracy. Section 6.2 begins with an analysis of the effect of the initial mesh resolution on the clustering process (6.2.1) and a comparison between static and dynamic p -adaptation strategies (6.2.2). Subsequently, 6.2.3 examines the results of static adaptation, focusing on three test cases: steady flow past a cylinder at $Re = 40$, unsteady laminar flow at $Re = 100$, and turbulent flow at $Re = 3900$. These test cases highlight the method's capability to balance computational efficiency and accuracy.

Finally, 6.2.4 presents a numerical experiment simulating the wake of a wind turbine using experimental blind test data from NTNU. This experiment demonstrates the

practical applicability of the GMM-assisted p -adaptation approach in a real-world engineering context, showcasing its ability to reduce computational effort while maintaining high fidelity in flow predictions.

6.1. Methodology for GMM-Based Mesh Refinement

This section details the methodology for implementing mesh adaptation driven by the Gaussian Mixture Model (GMM). This approach partitions the computational domain into distinct flow regions: a viscous/turbulent region (boundary layer and wake) and an inviscid/irrotational region (outer flow). By eliminating the need for user-defined thresholds, the GMM-based method improves robustness and adaptability across a range of flow regimes. The mesh adaptation process begins with an initial simulation to ensure the flow is fully developed, followed by clustering to identify regions requiring refinement. Using the identified clusters, polynomial orders are assigned adaptively, with high-order polynomials applied in viscous/turbulent regions and low-order polynomials in inviscid regions. This methodology optimizes computational resources while maintaining solution accuracy, as demonstrated through static and dynamic adaptation strategies for steady and unsteady flows.

The Gaussian Mixture Model (GMM), utilized in conjunction with the feature space described in [84], operates without the need for any parameters or thresholds, thereby eliminating the requirement for fine-tuning. Through a comparative analysis with traditional feature-based sensors (namely, the Q-criterion [47] and the new Ω method [67]), our aim is to underscore the advantages and enhancements offered by our approach. By removing problem-dependent threshold values, our sensor's applicability and robustness are improved, resulting in a more efficient and reliable mesh adaptation technique.

The proposal to utilize a high-order polynomial approximation for the viscous-dominated region and a low-order polynomial approximation for the inviscid region comes from the recognition that the former necessitates a finer mesh than the latter. This approach aims to more accurately predict flow features while achieving a solution equivalent to that obtained with a uniformly high polynomial order at a reduced cost. Notably, the advantage into only two regions reduces the computational overhead associated with mortar computation (compared to other techniques employing

different polynomial orders for each element in the domain [97, 55, 56]).

Furthermore, the clustering-adaptation process is executed only once the flow has developed, applicable to both steady and unsteady simulations. This simplifies the load-balancing process necessary for parallel computations. In the particular case of the high-order discontinuous Galerkin spectral element method (DGSEM), the mean of the membership probabilities $\bar{p}_v$ and $\bar{p}_i$ within each element is computed as follows, where $N = (P + 1)^3$ is the number of DoF of each element of the mesh, and P denotes the polynomial order associated with the high-order discretization:

$$\bar{p}_v = \frac{1}{N} \sum_{j=1}^N p_{v,j}, \quad p_{v,j} \in [0, 1], \quad (6.1)$$

$$\bar{p}_i = \frac{1}{N} \sum_{j=1}^N p_{i,j}, \quad p_{i,j} \in [0, 1]. \quad (6.2)$$

Each element in the mesh is then assigned two membership probabilities: p_v for the clustered viscous/turbulent region and p_i for the outer region. An element is assigned to the region with the highest mean of membership probabilities $\bar{p} = \max(\bar{p}_i, \bar{p}_v)$. The same process is applied for all elements to supply each one with a region ID, viscous/turbulent or inviscid.

The number of normal components (clusters) is determined as two, as the objective is to identify two distinct flow regions: a viscous/turbulent-dominated rotational region (boundary layer and wake region) and an inviscid irrotational region (outer region).

Furthermore, without loss of generality, an arithmetic mean is utilized to compute the mean of the membership probabilities. However, any numerical quadrature could have been considered to evaluate the element integral defining the average. It's important to note that the impact of the quadrature rule, used to categorize elements with mixed contributions inside, is minimal. Its effect is primarily noticeable on a small subset of elements located at the verge between inviscid and viscous/turbulent regions.

Two mesh adaptation strategies will be performed, static and dynamic p -adaptation. The dynamic adaptation is performed only with one test case of unsteady laminar flow to test its efficiency. We limited this strategy to this test case because it represents an unsteady laminar flow past a circular cylinder at a Reynolds number of 100. This case

is not computationally expensive and provides a good basis for evaluating whether static or dynamic p -adaptation is more suitable for our scenarios. To perform the cluster-based adaptation process, several sequential steps are undertaken. Initially, a simulation is executed with a specified polynomial order, denoted as $P_{initial}$, until the flow reaches a fully developed state. This choice of $P_{initial}$ ensures that the solver adequately captures all flow features with the desired level of accuracy.

Following the initial simulation, the clustering procedure is executed, using the $\{Q_S, R_S, Q_\Omega\}$ distribution as explained in the preceding redmention the chapter of clustering inviscid/viscous). It is imperative to select a sufficiently high value of $P_{initial}$ to ensure accurate capture of the invariants required for clustering. The influence of the initial mesh resolution on the clustered region is further analyzed in Section 5.2.2.

According to the approach outlined in [84], a single snapshot suffices for calculating the clustered region, applicable for both steady and unsteady problems, because adding multiple snapshots does not significantly change the identified viscous/turbulent flow regions.

The computational overhead for the clustering step is much lower compared to a single explicit iteration of the time stepper. Since adaptation is a one-time process, the clustering overhead is negligible.

Subsequently, based on the determined region ID for each element, the corresponding polynomial order within each element is assigned. In this study, a lower polynomial order, denoted $P_{inviscid}$, is applied in the inviscid regions, while a higher polynomial order, labeled $P_{cluster}$, is assigned to elements belonging to the viscous/turbulent region.

Taking practical considerations into account, the DGSEM integrated into the numerical framework, HORSES3D (outlined in previous sections), demonstrates optimal performance with moderately high polynomial orders, typically around $P = 4$ or fifth order. This balance between computational cost and accuracy is achieved at a fifth-order polynomial, presenting an effective compromise. As a result, we generate an h -mesh that attains high accuracy at $P = 4$. Subsequently, we apply the p -adaptation

methodology to selectively utilize $P = 4$ in the necessary regions ($P_{cluster} = 4$) In the inviscid regions, it is advisable to start with the minimum polynomial order $P_{inviscid} = 1$, and increase it only if numerical instabilities arise. Achieving the final ideal mesh involves an iterative process where $P_{cluster}$ can be incrementally raised until the accuracy requirements are fulfilled. This approach allows for selective augmentation of the polynomial order solely within a subdomain of the computational domain, thereby effectively reducing the overall simulation cost.

Alternatively, this methodology can be complemented with more advanced techniques, such as error-based adaptation [97, 23, 5] or output-based adaptation [63, 26, 13]. These methods facilitate the prediction of the required mesh resolution for a given error level, albeit often with significantly higher computational costs.

The simulation is subsequently restarted using the adapted polynomial order from the same instant until the final solution is reached. This process is outlined in Algorithm 1.

Algorithm 1 GMM mesh-adaptation algorithm

```

Start the simulation using uniform polynomial order  $P_{initial}$ 
Stop the simulation when the flow is fully developed
Train GMM with feature space  $E$ 
Return  $p_v$  and  $p_i$  ▷ Membership probabilities for every point
for  $k$  in number of elements do
     $\bar{p}_i = \frac{1}{N} \sum_{j=1}^N p_{i,j}$ 
     $\bar{p}_v = \frac{1}{N} \sum_{j=1}^N p_{v,j}$ 
    if  $\bar{p}_v \geq \bar{p}_i$  then
         $P(k) \leftarrow P_{cluster}$  ▷ high-order polynomial in the viscous rotational region
    else if  $\bar{p}_v < \bar{p}_i$  then
         $P(k) \leftarrow P_{inviscid}$  ▷ low-order polynomial in the inviscid irrotational region
    end if
end for
Return  $P$ 
Restart the simulation using  $P$ 

```

6.2. Results and Discussion

The application of the p -adaptation methodology was carried out on the test cases outlined in Section 5.1.2. Initially, in 5.2.1, we conducted a comprehensive evaluation

of regions earmarked thorough examination of the impact of threshold values on region detection for traditional sensors.

Building upon these preliminary assessments, in this Section, we explore the influence of initial mesh resolution on the clustered regions, aiming to discern optimal meshing strategies for different flow conditions. Subsequently, the results of the adaptation process will be detailed in Section 6.2.3, where we will present a comprehensive analysis of the adapted meshes and their performance in accurately capturing the flow. Through meticulous examination and comparison of the results, we seek to elucidate the efficacy of our p -adaptation methodology in optimizing computational resources while maintaining high levels of accuracy in flow simulations.

6.2.1. Analysis of the effect of the initial mesh resolution

The adaptation methodology outlined in Section 6.1 entails conducting an initial simulation to compute the clustered region. This simulation is conducted with a polynomial order $P = P_{initial}$, which must be sufficiently high to accurately capture the flow invariants. In this section, we investigate the influence of $P_{initial}$ on the clustered region for three distinct test cases: flow past a circular cylinder at Reynolds numbers 40, 100, and 3900.

To explore this effect, initial simulations are conducted with varying values of $P_{initial}$. Specifically, for laminar test cases ($Re = 40$ and 100), simulations are performed with $P_{initial} = 2$ and $P_{initial} = 4$, whereas for the turbulent test case ($Re = 3900$), simulations are conducted with $P_{initial} = 3$ and $P_{initial} = 4$.

Figure 6.1 presents a comparison of the clustered regions obtained using different values of $P_{initial}$. Remarkably, the clustered regions and the adapted meshes exhibit striking similarity across the various polynomial orders evaluated. This observation suggests that the initial mesh resolution has a minimal impact on the resulting clustered regions for the test cases considered in this study.

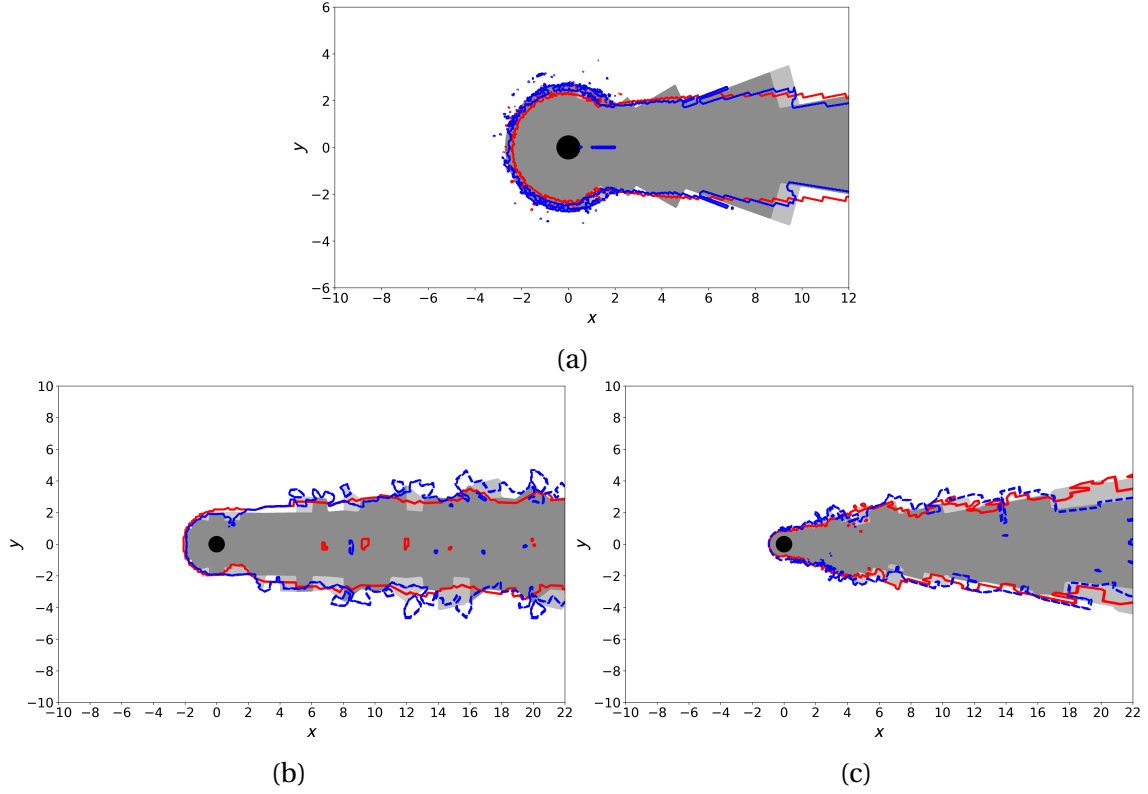


Figure 6.1.: Clustering and regions marked for adaptation when considering various polynomial orders to train the clustering GMM algorithm ($P_{initial} \in [2, 4]$ for $Re = 40$, $Re = 100$ and $P_{initial} \in [3, 4]$ for $Re = 3900$). Flow past a cylinder at $Re = 40$ (6.1a), $Re = 100$ (6.1b) and $Re = 3900$ (6.1c). **Red line:** Boundary of the viscous/turbulent region detected with GMM using $P = 4$. **Dashed blue line:** Boundary of the viscous/turbulent region detected with GMM using $P = 2$ for the flow at $Re = 40$, $Re = 100$ and $P = 3$ for the flow at $Re = 3900$. **Dark Grey:** Elements marked for adaptation when using either the red or blue clustered regions. **Light Grey:** Elements marked for adaptation which are selected only by one of the clustered regions (no overlapping).

6.2.2. Static Vs Dynamic mesh adaptation

In unsteady flow simulations, two distinct adaptation strategies emerge: dynamic and static adaptation, both extensively employed in the field. Dynamic adaptation methods, as illustrated in [8, 14], dynamically adjust polynomial degrees in response to evolving flow conditions. On the other hand, static adaptation methods, as exemplified in [78, 32], involve adjusting the polynomial degrees once throughout the simulation.

The dynamic p -adaptation method, a prominent example of dynamic adaptation [97], periodically updates polynomial degrees in unsteady flow simulations to accommodate changing flow characteristics. This strategy is particularly advantageous for transient simulations, where the region of interest evolves over time. The interval between adaptation stages can be specified in various ways, such as physical time or number of iterations. However, this method introduces overhead due to the need for reallocation of storage and projection of solutions in new polynomial spaces.

When incorporating clustering before the adaptation process, additional overhead is incurred from constructing clusters and adjusting polynomial orders based on the clustering results.

For our study, we will compare two adaptation strategies, static and dynamic adaptation approaches, to determine which method offers greater advantages in achieving the required accuracy with minimal computational overhead. By contrasting the outcomes of these approaches, our goal is to provide insights into the most efficient adaptation strategies tailored to specific flow scenarios.

In the static adaptation, the clustering methodology is performed only once when the wake is fully developed. Other than the static adaptation, in the dynamic adaptation, the clustering is performed periodically during a simulation and the polynomial degree is adapted immediately after every clustering. The polynomial order is then adjusted according to the clustering results every twenty iterations, selectively employing $P = 4$ in the viscous region ($P_{cluster} = 4$) and $P = 1$ ($P_{inviscid} = 1$) in the inviscid region. This iterative process resembles the one delineated in Algorithm 1, but it is executed repetitively instead of just once. With this comprehensive setup, we proceed to assess the performance of our novel sensor with an unsteady laminar flow at $Re = 100$.

Table 6.1 illustrates the computational time, encompassing adaptation and clustering time, for both static and dynamic p -adaptation approaches. A discernible observation

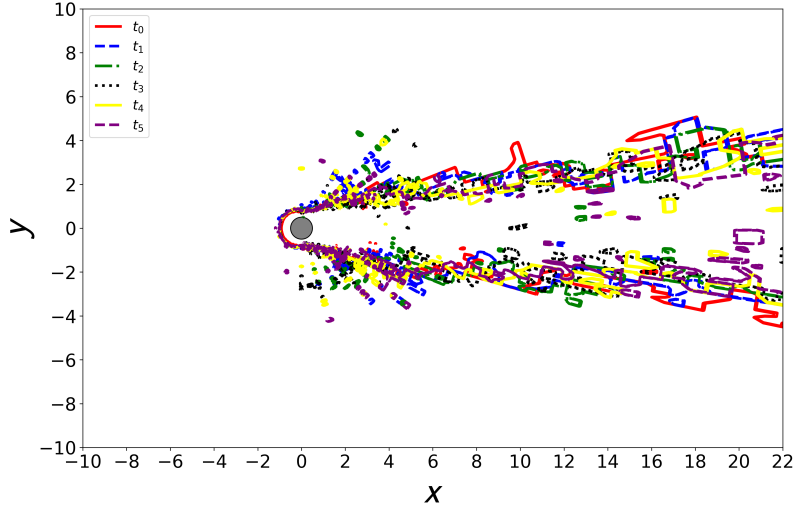


Figure 6.2.: Flow past a cylinder at $Re = 3900$: Flow region clustering at different time instants using GMM clustering with feature space E

from the table indicates that static adaptation entails lower costs compared to dynamic adaptation.

Table 6.1.: Comparison between static and dynamic adaptation for the flow past a cylinder at $Re = 100$

	Clustering Time (s)	Adaptation Time (s)	Total Time (s)
Static Adaptation	2.737×10^{-6}	1.157×10^{-5}	1.431×10^{-5}
Dynamic Adaptation	5.033×10^{-3}	8.066×10^{-3}	1.309×10^{-2}

Figure 6.2 represents the clustering of turbulent flow in $Re = 3900$ using six equidistant snapshots at different time instances t_i within a flow time period $T = 5s$ ($St = 0.2$). The similarity observed among all clusters indicates that they detect comparable viscous/turbulent flow regions. Hence, we can infer that a single snapshot adequately represents the wake behavior for an entire vortex shedding period.

In conclusion, for all test cases considered in this study, we opt for static adaptation, employing a single snapshot of the fully developed flow. This decision is based on the evaluation of multiple snapshots conducted in [84], which revealed no significant differences in the clustered regions. Therefore, in line with the methodology outlined in the same study, we rely on static adaptation for its practicality and consistency across various flow conditions.

6.2.3. Static Adaptation Results

In the preceding section, we demonstrated the advantages of the new clustering technique over traditional feature-based sensors. As we draw this study to a close, we will use the adaptation regions identified through the clustering technique, as discussed in Section 6.2.1, to execute mesh adaptation in the three test cases under consideration.

By employing the adaptation regions identified through the clustering technique, our objective is to capitalize on the enhanced accuracy and efficiency afforded by this methodology and to ensure that our mesh adaptation strategy is informed by comprehensive data-driven insights into the flow thus facilitating a more precise and targeted refinement of the computational mesh.

6.2.3.1. Flow past a cylinder at $Re = 40$

The adaptation process follows the procedure outlined in Algorithm 1. Figure 6.3a illustrates the clustered region and the adapted mesh resulting from this procedure. Notably, a high polynomial order is employed in the selected based on the mean probability of belonging to the viscous rotational region.

In this numerical experiment of the flow past a cylinder at $Re = 40$, a polynomial order of $P_{cluster} = 4$ is used for the viscous regions, while $P_{inviscid} \in [1, 2, 3]$ for the inviscid regions.

A visualization of the flow field is provided in Figure 6.3b, showcasing the norm of the velocity vector for the adapted solution ($P_{cluster} = 4, P_{inviscid} = 1$).

We run all simulations until the maximum residuals decrease by a factor of 1000. At this stage, while the flow was not converged, the wake had developed sufficiently. Using these data, we perform clustering to detect regions requiring adaptation or coarsening. Following adaptation, we continued the simulations until the maximum residual reached 10^{-4} , achieving a higher level of convergence.

This adapt-and-converge technique significantly reduced computational time while maintaining accuracy. Figure 6.4 illustrates the convergence history of both the homogeneous $P = 4$ solution and the clustering-adapted solution ($P_{cluster} = 4$ and $P_{inviscid} \in [1, 2, 3]$). It is noticeable that convergence speeds up with the implemen-

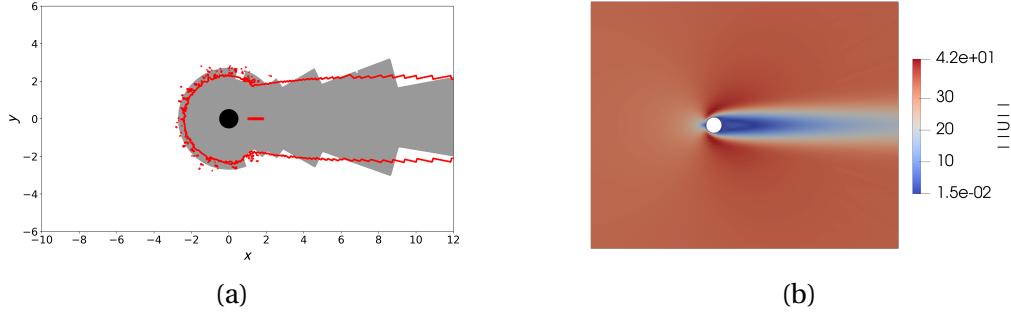


Figure 6.3.: Figure 6.3a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 40$. **Grey:** $P_{cluster} = 4$, **White:** $P_{inviscid} \in [1, 2, 3]$, **Red line:** Boundary of the viscous dominated, rotational region detected with GMM. Figure 6.3b presents the norm of velocity for the adapted flow field.

tation of adaptation. Throughout all simulations, a constant CFL number of 0.4 was maintained to ensure stability. The computational savings achieved and the corresponding Degrees of Freedom (DoF) for each simulation are presented in Table 6.2.

The adapted solutions for $P_{inviscid} \in \{1, 2, 3\}$ exhibited reductions in computational time by 32%, 29%, and 19% respectively compared to the homogeneous ($P = 4$) solution, with corresponding reductions in DoF by 67%, 55%, and 35%.

It's worth noting that the uniform $P = 1$ solution, represented with crosses in Table 6.2, did not converge due to insufficient resolution. This underscores the importance of adequate mesh refinement for accurate convergence and solution stability.

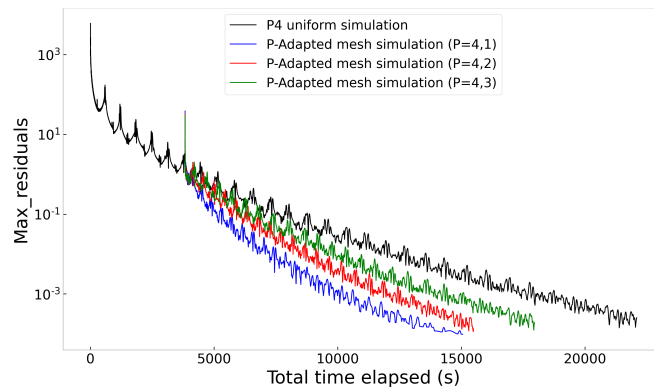


Figure 6.4.: Max residuals (the infinity norm of the temporal derivative in the semi-discrete Navier-Stokes equations) for homogeneous and p -adapted solutions

Finally, Table 6.3 presents drag values extracted from the literature for reference. By

Table 6.2.: Comparison of the degrees of freedom (DoF), computational cost and drag coefficient (C_d) for the uniform and adapted solutions for the steady flow past a circular cylinder at $Re = 40$

$P_{cluster}$	$P_{inviscid}$	Adapted	DoF	Comp. Time (s)	Reduction		C_d
					DoF	Comp. Time	
1	1	No	5472				
2	2	No	18468	3.78×10^3	78%	83%	1.5062
3	3	No	43776	7.90×10^3	49%	64%	1.5264
4	4	No	85500	2.21×10^4	-	-	1.5221
4	1	Yes	27936	1.50×10^4	67%	32%	1.5214
4	2	Yes	38460	1.58×10^4	55%	29%	1.5225
4	3	Yes	55488	1.79×10^4	35%	19%	1.5222

comparing Tables 6.3 and 6.2, we see that both the baseline homogeneous solution (for $P = 4$) and all clustered adapted solutions ($P_{cluster} = 4$ and $P_{inviscid} \in [1, 2, 3]$) yield accurate predictions for the drag coefficient.

This observation underscores the effectiveness of the clustering-based adaptation methodology in achieving reliable and accurate results comparable to those reported in the literature. By refining the computational mesh based on flow features, the adapted solutions demonstrate enhanced predictive capability while significantly reducing the computational costs.

The agreement between the predicted drag coefficients and the literature values further validates the robustness and applicability of the proposed adaptation approach in accurately capturing complex flow phenomena and predicting relevant aerodynamic parameters.

Table 6.3.: Comparison of the drag coefficient (C_d) results for flow past a cylinder at $Re = 40$ against experimental data

Case	C_d
Dennis and Chang [25]	1.52
Fornberg [34]	1.50
Choi et al. [18]	1.49

Comparison of Q-criterion, new Ω method and cluster based adaptation

In our evaluation of the adapted meshes generated by the Q-criterion, the new Ω method, and the new clustering technique, we focused on the cylinder at $Re = 40$, as detailed previously. We analyzed the drag coefficient predicted with different meshes in Figure 6.5.

The representation in Figure 6.5 includes the drag coefficient obtained with meshes utilizing uniform polynomial orders $P = 2$, $P = 3$, and $P = 4$. Furthermore, we integrated the adapted solution obtained through the clustering technique, as detailed previously, which employs $P_{cluster} = 4$ and $P_{inviscid} = 1$.

The solution obtained through the clustering technique shows excellent agreement with the uniform $P = 4$ solution, which represents the optimal outcome. This observation underscores the effectiveness of the clustering-based adaptation approach in achieving results comparable to those obtained with uniform high-order meshes.

In refining meshes using the Q-criterion and the new Ω method, the determination of an appropriate threshold is crucial. We extensively explored the entire spectrum of potential threshold values, as explored in Section 5.2.1, thereby, various identified regions for adaptation. Using $P_{cluster} = 4$ and $P_{inviscid} = 1$ for simulations, the drag values obtained are shown in Figure 6.5.

Significantly, meshes with similar number of degrees of freedom produces notably inferior drag predictions in contrast to those achieved through the clustering technique. This observation is consistent with the findings detailed in Section 5.2.2, where we illustrated that, among meshes with identical number of degrees of freedom, both the Q-criterion and the new Ω method struggled to accurately identify elements experiencing substantial viscous effects for adaptation. Furthermore, obtaining a drag prediction on par with that of the clustering technique requires a number of degrees of freedom similar to that required for the uniform $P = 4$ solution.

In this initial steady-laminar case, we have demonstrated the effectiveness of GMM clustering in effectively separating the viscous region (comprising the boundary layer and wake) from the outer inviscid region. Through refinement of the clustered region and coarsening of the inviscid outer region, we have illustrated that the detected clustered region significantly contributes to errors in drag and refining it leads to increased accuracy in the simulations.

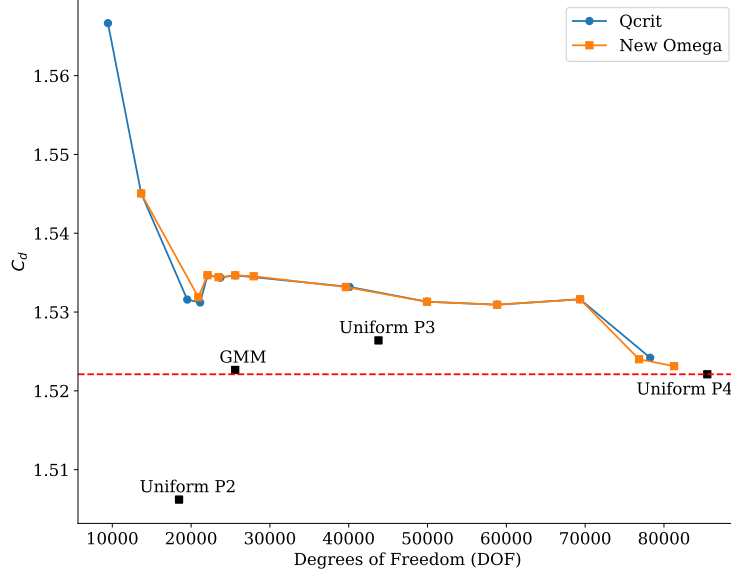


Figure 6.5.: Comparison of drag coefficients for flow past a cylinder at $Re = 40$. Individual data points represent uniform solutions at $P = 2$, $P = 3$, and $P = 4$, along with the GMM clustering-based technique employing $P_{cluster} = 4$ and $P_{inviscid} = 1$. The blue and orange lines illustrate solutions obtained with $P_{cluster} = 4$ and $P_{inviscid} = 1$, utilizing varying thresholds for the Q-criterion and new Ω method, respectively

Furthermore, our comparison of GMM clustering with traditional methodologies (Q-criterion and new Ω methods) highlights the superior efficiency of the new clustering methodology. This comparison underscores the importance of adopting advanced techniques for accurate flow characterization and mesh adaptation.

In conclusion, clustering adaptation emerges as a highly efficient technique for mesh adaptation in steady flows. Using data-driven insights to refine the computational mesh, this approach not only improves accuracy but also accelerates convergence, thereby improving the overall computational efficiency of the simulation process.

6.2.3.2. Flow past a cylinder at $Re = 100$

In our investigation of the clustering and mesh adaptation algorithm (Alg 1) for an unsteady-laminar flow past a cylinder at $Re = 100$, we apply the adaptation algorithm

to conduct p -adaptation based on a single snapshot of the fully developed flow.

The resulting domain, incorporating the adapted region, is depicted in Figure 6.6a. Additionally, Figure 6.6b showcases the velocity contour of the adapted solution with ($P_{cluster} = 4, P_{inviscid} = 1$).

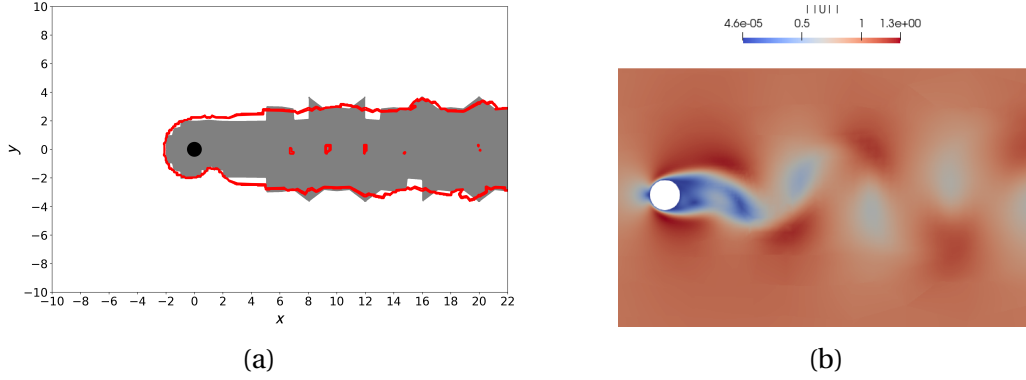


Figure 6.6.: Figure 6.6a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 100$. **Grey:** $P_{cluster} = 4$, **White:** $P_{inviscid} \in [1, 2, 3]$, **Red line:** Boundary of the viscous dominated, rotational region detected with GMM. Figure 6.6b presents the norm of velocity contour for the adapted case.

We run various simulations using uniform polynomial orders $P_{uniform} \in [1, 2, 3, 4]$, alongside different adapted solutions featuring $P_{cluster} = 4$ in the viscous regions and $P_{inviscid} \in [1, 2, 3]$ in the inviscid regions.

Throughout all simulations, a constant CFL value of 1.0 was consistently applied. For statistics, data collection began with a fixed time step dictated by the CFL condition. Statistical data were collected over an interval of $T = 150U_c/D$ non-dimensional time units, during which the average drag coefficient, the amplitude of the oscillations of the lift coefficient, and the Strouhal number were calculated. The resulting statistical values are summarized in Table 6.4, which shows the close resemblance of the adapted solutions to the uniform reference solution $P = 4$.

For a comprehensive comparison, Table 6.6 presents numerical results from the literature. A comparison between Tables 6.4 and 6.6 reveals the consistency of the simulations in this study with those reported in the literature. The proposed method effectively identifies regions of interest, enabling concentration of degrees of freedom in the viscous part while coarsening in the inviscid part. This approach allows

for the recreation of the reference solution at a lower cost with minimal error in unsteady functionals (lift, drag, and Strouhal number). In Table 6.5, an analysis of the

Table 6.4.: Average drag coefficient, amplitude of the lift coefficient oscillations and Strouhal number results for the unsteady flow past a circular cylinder at $Re = 100$.

$P_{cluster}$	$P_{inviscid}$	Adapted	C_d	C_l	St
1	1	No	1.2039 ± 0.0010	± 0.1243	0.1395
2	2	No	1.4050 ± 0.0075	± 0.3023	0.1623
3	3	No	1.3534 ± 0.0094	± 0.3214	0.1643
4	4	No	1.3468 ± 0.0096	± 0.3259	0.1642
4	1	Yes	1.3450 ± 0.0096	± 0.3245	0.1642
4	2	Yes	1.3468 ± 0.0096	± 0.3259	0.1642
4	3	Yes	1.3468 ± 0.0096	± 0.3259	0.1642

Degrees of Freedom (DoF) and computational cost for both uniform and adapted simulations is presented. It is evident that a simulation employing adaptation with ($P_{cluster} = 4, P_{inviscid} = 1$), exhibit smaller error compared to a uniform solution having $P = 3$, while utilizing fewer degrees of freedom and maintaining similar computational costs.

The observed differences in computational cost can be attributed to additional operations required for treating p -non-conforming elements and the larger timestep allowed by the CFL condition for uniform solutions with $P = 1$, as compared to $P = 4$ discretizations.

The adapted solutions, employing $P_{cluster} = 4$ and $P_{inviscid} \in [1, 2, 3]$, exhibit reductions in the number of degrees of freedom by 61%, 46%, and 26% respectively, compared to the reference $P = 4$ solution. Consequently, these adaptations can lead to reductions in computational cost by 20%, 18%, and 2% respectively.

The application of clustering adaptation to this second laminar case demonstrates its efficacy in handling simple unsteady problems. The developed algorithm adeptly identifies viscous regions using a single snapshot of the fully developed flow, facilitating the creation of an adapted mesh that accurately represents the solution with minimal error in the considered unsteady functionals.

Table 6.5.: Comparison of the degrees of freedom (DoF) and computational cost for the uniform and adapted solutions for the unsteady flow past a circular cylinder at $Re = 100$.

P_{cluster}	P_{inviscid}	Adapted	DoF	Comp. Time (s)	Reduction	
					DoF	Comp. Time
1	1	No	10256	4.90×10^2	84%	85%
2	2	No	23076	1.11×10^3	64%	66%
3	3	No	41024	2.12×10^3	36%	36%
4	4	No	64100	3.31×10^3	-	-
4	1	Yes	25082	2.66×10^3	61%	20%
4	2	Yes	34372	2.71×10^3	46%	18%
4	3	Yes	47378	3.24×10^3	26%	2%

Table 6.6.: Comparison of numerical results from the literature for the unsteady flow past a circular cylinder at $Re = 100$.

	C_d	C_l	St
Liu et al. [68]	1.350 ± 0.012	± 0.339	0.164
Calhoun [15]	1.330 ± 0.014	± 0.298	0.175
Braza et al. [10]	1.346 ± 0.015	± 0.250	0.160

6.2.3.3. Flow past a cylinder at $Re = 3900$

In our preceding test cases, we noted that the resolution in the inviscid region does not notably affect accuracy. Hence, in this particular test case of the flow past a cylinder at $Re = 3900$, we conducted computations using a uniform polynomial order ranging from $P \in [1, 2, 3, 4]$, alongside an adapted solution generated through the clustering algorithm with $P_{\text{cluster}} = 4$ and $P_{\text{inviscid}} = 2$.

In this turbulent scenario, employing $P_{\text{inviscid}} = 1$ led to divergent simulations, probably due to aliasing instabilities at the interface between elements with differing polynomial orders (mortar surfaces). Consequently, we only present results for $P_{\text{inviscid}} = 2$. The adaptation process involved utilizing the clustering algorithm on a single snapshot of the flow field after the wake had developed.

The resulting adapted mesh is visualized in Figure 6.7a, showcasing the refinement

achieved through the clustering algorithm in capturing the complex flow features associated with turbulent phenomena.

A constant CFL value of 0.5 has been uniformly maintained throughout all simulations. The time step for statistical data collection is fixed at the value prescribed by the CFL condition when the statistical data collection begins, ensuring consistency in data collection procedures.

To mitigate aliasing errors and enhance method robustness, a split-form discretization with Pirozzoli averaging [92] has been implemented.

For visualization purposes, Figure 6.7b provides a representation of the flow field for the adapted solution.

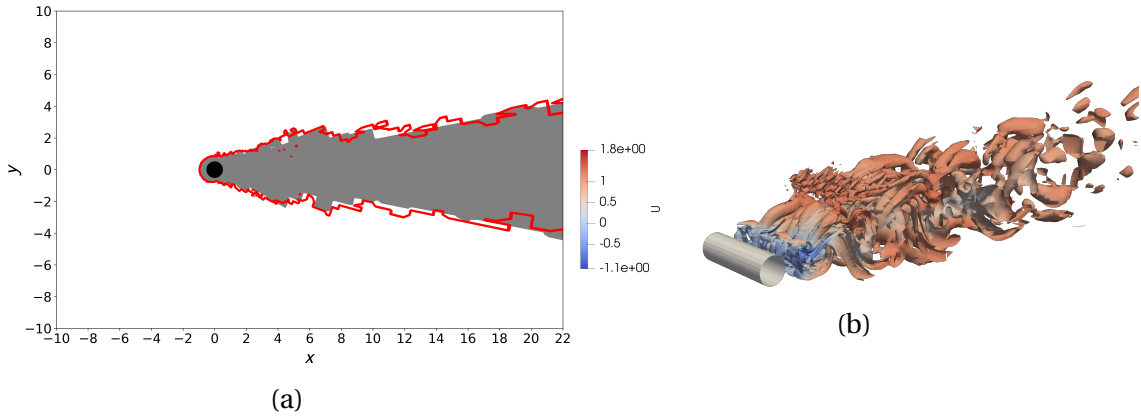


Figure 6.7.: Figure 6.7a presents the polynomial order distribution of the adapted mesh for the flow past a cylinder at $Re = 3900$. **Grey:** $P_{cluster} = 4$, **White:** $P_{inviscid} = 2$, **Red line:** Boundary of the viscous dominated, rotational/turbulent region detected with GMM. Figure 6.7b presents the contour of the spanwise velocity colored with the axial velocity u for the adapted solution.

In order to gather statistical quantities, we adhere to the methodology outlined in [87]. Initially, we allow the flow to develop for a time interval of $t = 150D/U_c$ to mitigate the influence of initial conditions. Subsequently, statistical data is collected over a duration of approximately 100 shedding cycles for each simulation, aligning with the averaging intervals commonly used in the literature [87].

The results pertaining to the mean drag coefficient C_d , Strouhal number St , and the recirculation length L_r are presented in Table 6.7. Notably, the lift coefficient C_l is not considered in this table, as it is not typically documented for this specific test case.

A comparison is drawn between the results obtained from HORSES3D simulations employing uniform polynomial orders $P \in [1, 2, 3, 4]$ and a p -adapted mesh, against both numerical and experimental results from prior studies. It is evident from the table that both the uniform $P = 4$ and adapted solutions fall well within the range of values reported in the literature.

However, for the uniform $P = 3$ solution, a significant under-prediction of the mean drag value is observed, attributed to insufficient resolution. Despite this, the Strouhal number remains relatively close to the reported values. Lower polynomial orders exhibit discrepancies in both mean drag value and Strouhal number, highlighting the importance of adequate resolution for accurately capturing flow dynamics.

Table 6.7.: Comparison of the mean statistical quantities for the cylinder at $Re = 3900$ using HORSES3D and literature data. We compare the Strouhal number St , the drag coefficient C_d , and the recirculation length L_r . All simulations used a spanwise length $L_z/D = \pi$.

	St	C_d	L_r/D	L_z/D
Uniform P1	0.172	2.2394	0.97	π
Uniform P2	0.219	1.1213	1.37	π
Uniform P3	0.202	0.7844	1.36	π
Uniform P4	0.203	0.9513	1.64	π
Cluster-Adapt P4-P2	0.204	0.9506	1.63	π
Parnadeau et al. [87]	0.208	-	1.56	π
Snyder and Degrez [110]	0.207	1.09	1.30	π
Kravchenko and Moin [60]	0.210	1.04	1.35	π
Breuer [11]	-	1.07	1.20	π
Franke and Frank [35]	0.209	0.98	1.64	π
DNS Ma et al. [71]	0.219	1.59	-	π
Ouvrard et al. [85]	0.223	0.94	1.56	π

The clustering algorithm demonstrates its capability to effectively track the region of interest, enabling local p -refinement and yielding a solution with 41% fewer Degrees of Freedom (DoF) while maintaining a similar level of accuracy compared to the uniform $P = 4$ solution, as detailed in Table 6.8.

Moreover, the adapted solution exhibits only a 16% increase in DoF compared to

the uniform $P = 3$ solution, but offers superior results, as illustrated in Table 6.7 and Figure 6.8.

Regarding computational cost, the adapted solution is 33% faster than the uniform reference solution $P = 4$ for any given simulation interval as indicated in Table 6.8. This efficiency gain underscores the effectiveness of the clustering algorithm in achieving computational savings without compromising solution accuracy.

Table 6.8.: Comparison of DoFs and computational time for the cylinder at $Re = 3900$

	DoFs	Reduction of DoFs	Reduction of Comp. Time
Uniform P1	0.17M	93%	94%
Uniform P2	0.56M	78%	82%
Uniform P3	1.33M	49%	56%
Uniform P4	2.60M	-	-
Cluster-Adapt P4-P2	1.55M	41%	33%

The statistical results for this test case are presented in Figure 6.8, offering a comprehensive comparison with experimental (PIV) and numerical (LES) data from [87]. Notably, the uniform $P = 1$ results exhibit significant disparities compared to both numerical and experimental results due to inadequate resolution, thus these results are omitted from further analysis.

In Figure 6.8a, the variance of the streamwise fluctuations at three different positions $x/D = [1.06, 1.54, 2.02]$ is depicted. Both the adapted and uniform $P = 4$ solutions demonstrate excellent agreement with the PIV and LES data. Notably, the transitional state of the shear layer is evident, particularly at the position $x/D = 1.06$, where two distinct peaks emerge, which are noticeably diminished in the uniform $P = 3$ case, and nearly vanish in the $P = 2$ case, indicative of insufficient resolution [87].

Figure 6.8b presents the covariance of the fluctuations, where again, the adapted and uniform $P = 4$ solutions align well with the reference data, while deviations are observed for the uniform $P = 2, 3$ solutions, highlighting the insufficiency of resolution to capture the correct phenomena at these levels.

Figures 6.8d and 6.8e depict the mean streamwise and normal velocities, with less prominent differences, the results of this study show good agreement with the reference results (LES, PIV). The variance of the normal velocity fluctuations in Figure 6.8f

also aligns well, although both the uniform and clustering p -adaptation solutions underestimate the peak at $x/D = 2.02$.

Finally, Figure 6.8c presents the mean streamwise velocity along the centerline, where the results closely match the reference data, albeit with a slight over-prediction of the point of minimum velocity observed in the HORSES3D results.

The exploration of this third turbulent case underscores the efficacy of clustering adaptation in addressing turbulent unsteady scenarios. Using GMM clustering, the algorithm adeptly identifies viscous and turbulent regions within the boundary layer and wake, where higher resolution is imperative. Currently, it accurately separates these areas from the outer inviscid region, where a lower resolution suffices. This adaptive approach not only ensures computational efficiency but also enhances solution accuracy by appropriately allocating computational resources based on flow features.

6.2.4. Numerical experiments

This section embarks on a numerical experiment aimed at illustrating the practicality and efficacy of the developed machine learning adaptation methodology within an engineering context. Specifically, we simulate the experimental blind test (BT) conducted by the Norwegian University of Science and Technology (NTNU). The BT involves a three-blade wind turbine with a diameter (D) of 0.894 m, tested in the wind tunnel of the Department of Energy and Process Engineering, NTNU. The turbine blades were made of NREL S826 airfoils [61].

The wind tunnel used for the experiment had dimensions of $[L \times W \times H] = [11.15 \times 2.71 \times 1.8]$ m. It is pertinent to note that the inflow velocity profile was uniform throughout the experiment, while turbulence intensity levels were maintained low at 0.3%.

In our analysis, we used the experimental setup and data gleaned from the blind test, which gives insights into the wake of a solitary wind turbine across varying tip speed ratios (λ). Our focus is on the outcomes obtained at the optimal tip speed ratio $\lambda = \Omega D / 2U_{\text{ref}} = 6$, where $U_{\text{ref}} = 10$ m/s and $\Omega = 134.228$ rad/s, testing the efficiency of our adaptation methodology. Notably, the blade tip Reynolds number for this particular scenario is $Re_c = \lambda U_{\text{ref}} c_{\text{tip}} / \nu = 103600$, with $c_{\text{tip}} = 0.025926$ m representing the tip chord length, and ν denoting the kinematic viscosity of air.

To model the turbine blades' behavior in our computational model, we employ a dual approach. Firstly, we utilize the immersed boundary method [29, 59] to represent the tower and the nacelle. Concurrently, the rotating blades are represented using an actuator line method [111], ensuring a comprehensive and accurate depiction of the wind turbine's aerodynamic interactions within the flow domain.

For our simulations, we employ the Large Eddy Simulation (LES) computational fluid dynamics (CFD) technique, solving the compressible 3D Navier-Stokes equations augmented with the Vreman turbulence closure model. The simulations are conducted using the HORSES3D software [29].

The wind tunnel boundaries are modeled as free slip walls, while no inlet turbulence is introduced into the simulation.

The computational mesh is Cartesian with an identical size to the wind tunnel, comprising $128 \times 24 \times 24$ elements. This mesh configuration ensures a spatial resolution commensurate with the intricacies of the flow field, with a $D/\Delta x$ ratio of approximately 10, where D represents the turbine diameter. It's worth noting that due to the utilization of a high-order method, the spatial resolution also improves as the polynomial order increases. Throughout the simulations, we maintain a constant CFL value of 0.4 to uphold numerical stability and accuracy. To gather statistical insights, the time step remains fixed at a value dictated by the CFL condition when the statistical data collection begins.

Initially, we run four non-adapted simulations employing a uniform polynomial order of $P \in [1, 2, 3, 4]$. These simulations are run for 2 seconds, allowing for the wake to fully develop in the first second and it needs an additional second to stabilize. Afterward, statistics are gathered over a duration of 0.5 seconds, which corresponds to 10 rotations of the turbine blades.

Additionally, we conduct one adapted simulation using Algorithm 1. Initially, a simulation is conducted for one second to allow for the development of the wake. This simulation adopts a polynomial order of $P_{initial} = 4$ to ensure accurate representation of the wake. Subsequently, adaptation takes place by configuring $P_{cluster} = 4$ and $P_{inviscid} = 1$, continuing the simulation for an additional second. Once more, we opt for the lowest feasible resolution in the inviscid region, setting $P_{inviscid} = 1$, as we have noted that its influence on the simulation's accuracy is minimal. Subsequently,

statistical computations are conducted over a duration of 0.5 seconds.

The adaptation process employs the clustering algorithm on a single snapshot of the flow field after the wake has developed. The adapted mesh is showcased in Figure 6.9a. A visual representation of the flow field for the adapted solution is depicted in Figure 6.9b, highlighting the clustering algorithm's adeptness in identifying the viscous/turbulent regions correctly.

In Figure 6.10, horizontal profiles of the mean streamwise velocity deficit are presented at various downstream positions ($x/D = 1, 3$, and 5), as depicted in vertical planes in Figure 6.9b. The velocity deficit for different solutions with uniform meshes $P \in [1, 2, 3, 4]$ is compared to experimental data [61]. As the polynomial order increases, there is a noticeable improvement in the solutions. However, despite this enhancement, a discernible discrepancy persists between the numerical solutions and experimental data, even at higher polynomial orders. This disparity is likely attributed to modeling errors inherent in representing the tower, nacelle (via the immersed boundary method), and rotating blades (utilizing the actuator line method). A similar phenomenon is documented in [130]. Nonetheless, of greater significance for this study is the striking resemblance between the adapted simulation and the homogeneous $P = 4$ solution, underscoring the efficacy of the clustering technique for intricate cases.

Table 6.9 provides a concise overview of the computational cost advantages associated with cluster adaptation. Using local p -refinement, we achieve a solution with 61% fewer degrees of freedom (DoF) and 43% faster computation, all while preserving a level of accuracy similar to the uniform $P = 4$ solution. This numerical experiment underscores the utility of clustering adaptation within the realm of general engineering applications, such as wind turbine simulations. The GMM clustering method adeptly delineates viscous and turbulent regions within the turbine wake, necessitating higher resolution, while effectively discerning them from the outer inviscid region, where lower resolution suffices. This strategic approach optimizes computational resources while ensuring a faithful representation of the flow characteristics.

Table 6.9.: Comparison of DoFs and computational time for the turbine test case.

	DoFs	Reduction of DoFs	Reduction of Comp. Time
Uniform P1	0.59M	93.6%	92.7%
Uniform P2	1.99M	78.2%	86.5%
Uniform P3	4.72M	48.7%	54.1%
Uniform P4	9.22M	-	-
Cluster-Adapt P4-P1	3.58M	61%	43%

6.3. Conclusion

In this chapter, we explored the application of a Gaussian Mixture Model (GMM)-based methodology for p -adaptation in computational fluid dynamics (CFD) simulations. By partitioning the flow field into viscous/turbulent and inviscid/irrotational regions, the GMM-driven approach enabled efficient and accurate mesh refinement without the need for predefined thresholds. The proposed methodology was systematically evaluated through a series of numerical experiments.

The results demonstrated the robustness of the GMM-based clustering in effectively detecting flow features and guiding p -adaptation strategies. Both static and dynamic adaptation approaches were evaluated. While dynamic p -adaptation dynamically adjusts polynomial degrees during the simulation to handle evolving flow conditions, it incurs significant computational overhead due to the need for repeated clustering and polynomial adjustments. On the other hand, static p -adaptation, where clustering is performed only once after the flow is fully developed, emerged as the more computationally efficient strategy. This approach maintains accuracy while significantly reducing computational costs, as evidenced by its performance across various flow scenarios. The evaluation also confirmed that using a single snapshot of the fully developed flow is sufficient for clustering, ensuring practical applicability and consistency without compromising the quality of the adaptation process. Additionally, the methodology was successfully applied to a wind turbine wake simulation using experimental data from NTNU's blind test. The GMM-assisted p -adaptation approach demonstrated its potential in real-world engineering applications by significantly reducing computational costs while maintaining high fidelity in capturing the complex flow physics.

These findings underscore the versatility and efficiency of GMM-based p -adaptation for adaptive CFD simulations. Using machine learning techniques to enhance mesh refinement strategies, this methodology offers a robust framework to improve simulation accuracy and efficiency in diverse flow scenarios.

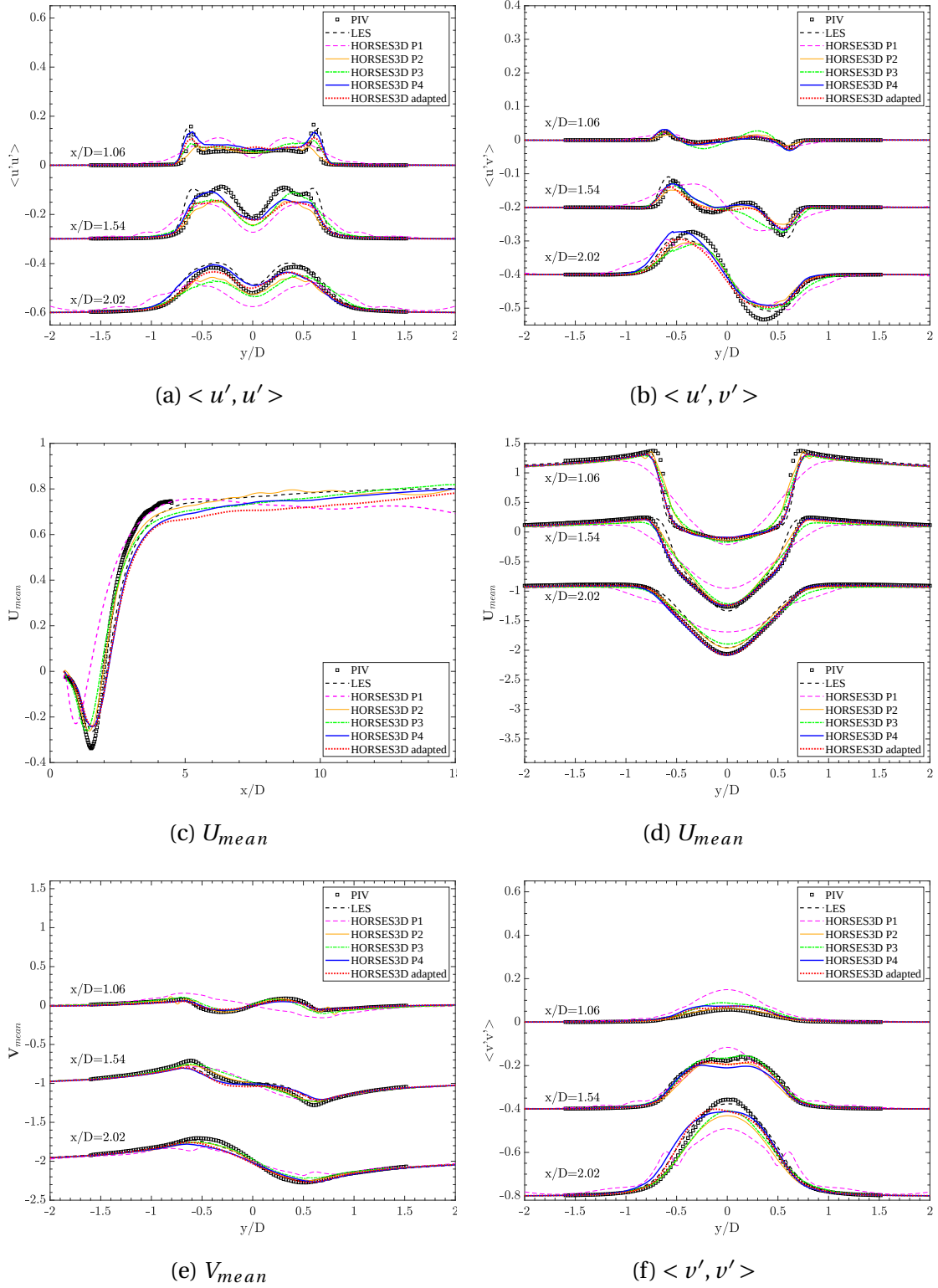


Figure 6.8.: Statistical quantities of the wake for the unsteady turbulent case of a flow past a cylinder at $Re = 3900$. Comparison of experimental (PIV) and high resolution numerical (LES) results from [87] against uniform and adapted results from HORSES3D

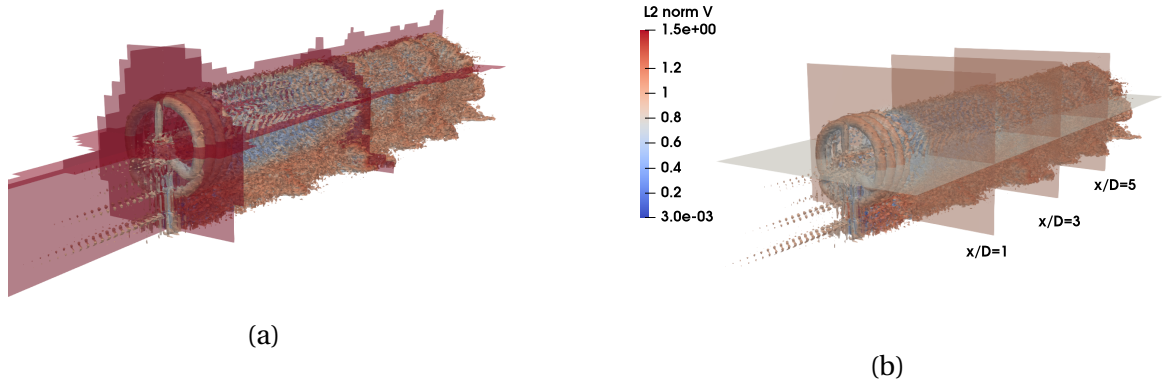


Figure 6.9.: Contours of the Q-criterion, with colors representing the L2 norm of the velocity, for the flow past a wind turbine at $Re_c = 103600$. Figure 6.9a also includes slices of the detected viscous dominated, rotational/turbulent region, highlighted in red, using the Gaussian Mixture Model (GMM) analysis. Figure 6.9b displays both horizontal and vertical planes. The intersection of these planes marks the positions where the velocity deficit at the horizontal axis is compared with experimental data.

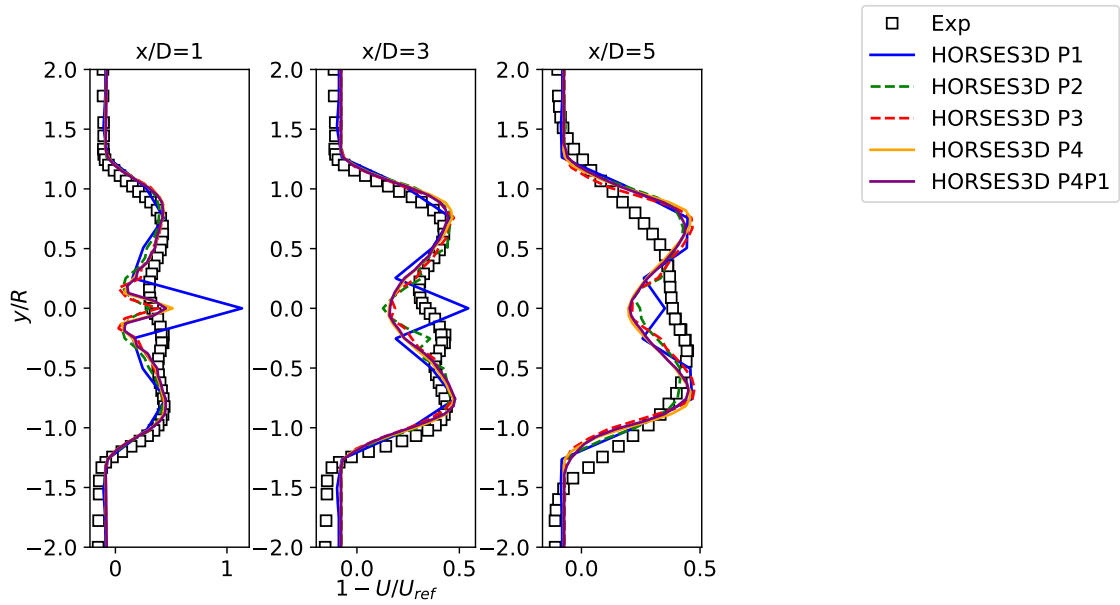


Figure 6.10.: Horizontal profiles of mean streamwise velocity deficit at three downstream positions x/D for $P \in [1, 2, 3, 4]$ and adapted ($P_{cluster} = 4$ and $P_{inviscid} = 1$) and comparison with experimental data [61].

Chapter 7

Conclusions

This thesis has explored the integration of machine learning techniques, specifically clustering algorithms, into Computational Fluid Dynamics (CFD) simulations. The primary objective was to enhance the accuracy and computational efficiency of high-order numerical methods through p -adaptation using clustering-based feature detection.

7.1. Summary of the work

The work was structured into two parts: the first established the theoretical and methodological background, while the second presented novel contributions and experimental results.

In the first part of the thesis (chapters 2 and 3) we covered the background and foundations for this project where chapter 2 introduced CFD principles, governing equations, and discretization methods, highlighting the trade-off between computational cost and accuracy. The importance of mesh adaptation was emphasized as a strategy for optimizing simulations. Chapter 3 provided an overview of machine learning, with a focus on unsupervised learning techniques such as K-means and Gaussian Mixture Models (GMM). The application of clustering for identifying flow features was introduced as a promising approach to adaptive mesh refinement.

Chapter 4 presented the computational framework, selecting the Discontinuous Galerkin Spectral Element Method (DGSEM) in HORSES3D as the high-fidelity solver.

The GMM was introduced as a flexible data-driven approach for clustering CFD data. Chapter 5 proposed a GMM-based feature detection method, offering an automated alternative to threshold-based techniques like the Q-criterion. The approach demonstrated improved robustness in separating viscous and inviscid regions. Chapter 6 developed a novel clustering-based p -adaptation method, demonstrating that static adaptation (based on a single fully developed flow snapshot) provides a computationally efficient balance between accuracy and cost. Validation with experimental data from the NTNU blind test confirmed its practical applicability.

7.2. Contributions and Achievements

This thesis has made several contributions to CFD and data-driven simulation methodologies:

- Development of a GMM-based p -adaptation framework that eliminates predefined thresholds, providing an adaptive strategy for high-order CFD simulations.
- Demonstration that static clustering is an efficient alternative to dynamic adaptation, reducing computational overhead while maintaining accuracy.
- Validation of the proposed methods in complex flow cases, including real experimental datasets, showcasing their practical applicability.

7.3. Future Work

Although the proposed methods provide a strong foundation for clustering-based p -adaptation in CFD, several directions remain open for future exploration and development:

- **Automated Polynomial Degree Selection:** One natural extension of this work is the automation of polynomial degree assignment in p -adaptation. By incorporating additional flow-based metrics, such as the truncation error, into the feature space of the machine learning model, it may be possible to infer more accurate and physically informed degree distributions.
- **Enhanced Feature Spaces:** The current methodology relies primarily on velocity

gradient-based quantities for clustering. Future efforts could explore expanding the feature space to include other physically relevant indicators, such as pressure gradients, entropy generation, or local flow topology metrics, to improve clustering robustness and adaptivity.

- **Alternative Machine Learning Models:** Although GMM has shown promising results, it would be valuable to explore alternative or more advanced machine learning models, such as decision trees, auto-encoders, or neural networks, for mesh adaptation. These models could potentially handle more complex non-linear relationships and facilitate adaptive strategies beyond simple clustering.
- **Hybrid Adaptation Techniques:** Combining clustering-driven p -adaptation with other adaptive strategies (e.g. h -adaptation or time-step adaptation) could lead to more versatile and efficient simulation frameworks, particularly for transient or multiscale flows.
- **Real-Time and Online Adaptation:** Implementing low-overhead models capable of online learning could enable real-time mesh adaptation for industrial applications, particularly in the context of live simulation environments.

7.4. Final Remarks

In conclusion, this thesis has demonstrated the viability and potential of clustering-based machine learning techniques to enhance high-order CFD simulations. The integration of data-driven methods, such as GMM clustering, provides a flexible and robust framework for feature detection and adaptive refinement, helping to address key challenges in computational cost and accuracy.

The methodologies presented here are readily extensible and form a basis for continued research in the fusion of numerical methods and machine learning. Future work may enable increasingly autonomous, accurate, and efficient CFD simulations for a wide range of engineering applications using the growing capabilities of data-driven algorithms in greater depth.

Appendix A

Shock Detection using GMM

Shock capturing remains a critical challenge in computational fluid dynamics, particularly in accurately simulating complex flow phenomena characterized by shock waves. Traditional numerical methods often struggle to capture shocks with the required fidelity, leading to inaccuracies and computational inefficiencies. To overcome these limitations, unsupervised machine learning techniques have emerged as powerful tools for shock detection, offering data-driven and adaptive capabilities. Among these, Gaussian Mixture Models (GMMs) provide a probabilistic clustering approach that effectively identifies shocks by distinguishing regions based on flow gradients. Compared to traditional approaches, GMM-based methods introduce a self-adaptive mechanism that reduces user-defined tuning parameters while maintaining high accuracy in detecting discontinuities. Recent studies have explored the integration of machine learning methods in CFD, particularly for shock detection. In [88] Patel and Kushwaha demonstrated that GMM outperforms K-means in capturing complex discontinuity structures. Similarly, Zhu et al. in [128] applied clustering techniques to identify shock structures in one- and two-dimensional flows, showing the effectiveness of unsupervised learning in numerical simulations. These approaches align with broader trends in CFD, where data-driven methodologies are increasingly being used for sensor development, feature extraction, and numerical stabilization.

This appendix presents an application of GMM-based shock detection for high-order CFD simulations. The study focuses on analyzing shock regions, evaluating clustering performance using statistical metrics such as the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), and visualizing the clustering results. The

primary goal is to assess the robustness and effectiveness of GMM clustering compared to classical shock sensors. The appendix is structured as follows: [A.1](#) describes the test cases analyzed, including both inviscid and viscous flows. [A.2](#) provides an overview of the GMM-based shock detection methodology, including feature selection and model selection using AIC/BIC criteria. Finally, [A.3](#) compares the GMM-based approach to traditional shock sensors, presenting qualitative and quantitative assessments.

Note: The numerical simulations in this appendix were not part of my work. They were generated in a collaborative study and the GMM-based shock detection was applied to the data afterward.

A.1. Description of the Test Cases

Shock detection methodologies must be robust across different flow conditions, particularly in both inviscid and viscous flows. Two distinct test cases from the shock capturing study are considered: Inviscid flow around a cylinder at Mach 3. A purely hyperbolic problem without viscous effects, where strong discontinuities and shock waves dominate the flow and a Viscous flow around a cylinder at Mach 2, a more complex case where boundary layers influence shock interactions, requiring numerical stabilization. The performance of the Gaussian Mixture Model (GMM)-based shock detection is evaluated in these scenarios by analyzing shock localization, sensor response, and feature space clustering.

A.1.1. Inviscid Flow Around a Cylinder at Mach 3

The inviscid test case examines shock detection in a purely hyperbolic regime, where strong discontinuities emerge due to supersonic flow past a cylindrical obstacle. The absence of viscosity simplifies the governing equations but introduces challenges in capturing sharp discontinuities without excessive numerical dissipation.

A.1.2. Viscous Flow Around a Cylinder at Mach 2

The viscous test case introduces shock-boundary layer interactions with a Reynolds number of $Re = 10^5$. Unlike the inviscid case, viscosity leads to boundary layer forma-

tion, shear-layer interactions, and the smoothing of shock structures.

A.2. Gaussian Mixture Model (GMM) for Shock Detection

Accurate shock detection plays a crucial role in computational fluid dynamics, ensuring that numerical solvers correctly capture discontinuities while avoiding excessive dissipation in smooth regions. Traditional shock sensors rely on manually tuned thresholding methods, making them sensitive to specific flow configurations and parameter choices. To overcome these limitations, this study employs Gaussian Mixture Models as an unsupervised machine learning approach for shock detection.

A.2.1. Application of GMM to Shock Detection

The GMM is a probabilistic clustering algorithm that models the feature space as a mixture of Gaussian distributions. Unlike traditional shock sensors, GMM does not require predefined thresholds or explicit feature weighting, making it a flexible and adaptive approach. The probability density function (PDF) of the GMM is given by:

$$f(\mathbf{x}) = \sum_{j=1}^K \tau_j \mathcal{N}(\mathbf{x}; \mu_j, \Sigma_j), \quad (\text{A.1})$$

where K represents the number of clusters, τ_j is the weight of each Gaussian component, μ_j is the mean vector, and Σ_j is the covariance matrix.

In the context of shock detection, the GMM clusters the flow field into distinct regions, effectively separating smooth flow areas from shock-induced discontinuities. Unlike methods such as k -means clustering, which assume spherical clusters and hard assignments, GMM provides soft assignments, enabling a probabilistic measure of how likely each point belongs to a given cluster.

A.2.2. Feature Selection for GMM Clustering

The effectiveness of GMM in shock detection strongly depends on the selection of appropriate flow features. Since shocks are characterized by large gradients and abrupt changes in flow properties, the feature space is constructed using physically relevant quantities that highlight these discontinuities. The selected features include:

- **Velocity divergence** $(\nabla \cdot \mathbf{v})^2$: Identifies compression regions where shocks occur.
- **Pressure gradient magnitude** $\|\nabla p\|^2$: Captures pressure discontinuities across shocks.

Both features are normalized to the range $[0, 1]$ to ensure consistency in clustering results across different flow regimes.

A.2.3. Note on the number of clusters for the flow

In this subsection, we focus on a quantitative analysis to assess how the number of clusters influences the performance of the GMM-based sensor.

In discussing the number of clusters for the flow analysis using Gaussian Mixture Models (GMM), it is important to understand how different clusters affect the performance of the sensor, and how to determine the optimal number of clusters using statistical criteria like the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC). The use of these metrics help quantify how well the model fits the data and informs the decision on the number of clusters. These metrics balance the model's likelihood with its complexity (i.e., the number of free parameters) to avoid overfitting. The "BIC" is given by the equation:

$$\text{BIC} = -2 \log L + N_p \log N,$$

where: $\log L$ is the log-likelihood of the model. N_p represents the number of free parameters in the model. N is the number of data points in the feature space. For GMMs, the number of free parameters N_p is calculated as:

$$N_p = K + (K + 1)v + (K + 1)\frac{v(v + 1)}{2},$$

where: K is the number of clusters. v is the dimensionality of the feature space.

Another metric, the “AIC”, is defined similarly but with a different penalty term:

$$\text{AIC} = -2 \log L + 2N_p.$$

While both metrics penalize models based on their number of parameters, they differ in the strength of this penalty. In this context, the BIC tends to favor simpler models more than the AIC does, as the penalty grows with the number of data points N .

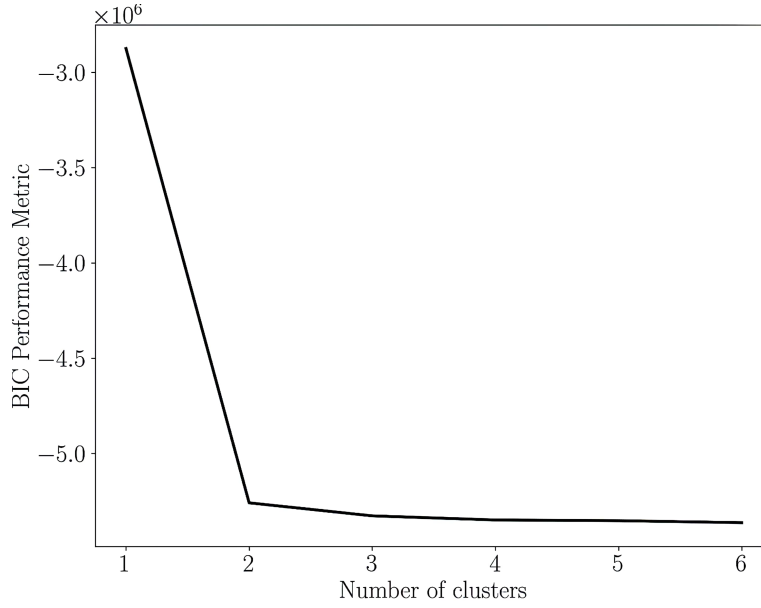


Figure A.1.: Bayesian information criterion with respect to the number of clusters.

Table A.1.: AIC and BIC values presented in Figure A.1.

Nº of clusters	AIC	BIC
1	-2,875,526	-2,875,474
2	-5,259,388	-5,259,274
3	-5,327,804	-5,327,627
4	-5,349,502	-5,349,263
5	-5,353,279	-5,352,977
6	-5,363,607	-5,363,243

In the analysis presented, the BIC values for different numbers of clusters are plotted and shown in Table A.1. The plot illustrates that the use of two clusters captures most

of the relevant information in the flow field. This conclusion follows from the BIC's "elbow point," where the rate of improvement in the BIC starts to diminish. Specifically, increasing the number of clusters beyond two yields diminishing returns, as indicated by the small changes in BIC values from three to six clusters.

The "elbow method" described above helps determine the optimal number of clusters, where additional clusters do not provide significant improvement and may risk overfitting. In the case of GMM for the shock sensor, this is evident when two clusters are sufficient to differentiate between smooth flow and discontinuities (i.e., shocks).

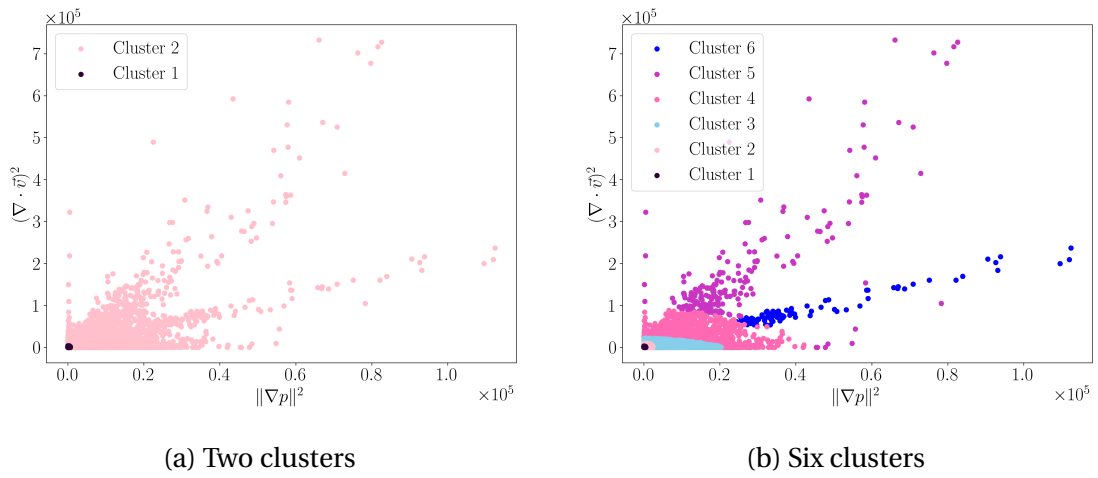


Figure A.2.: Feature-space clustering with the GMM sensor applied to the viscous case.

The clustering results for two and six clusters are visualized in Figure A.2. When two clusters are used, the first cluster represents most of the smooth flow regions, while the second cluster identifies nodes associated with shocks. Adding more clusters, as in the six-cluster case, allows for finer distinctions in the feature space, but the primary distinction between smooth and non-smooth regions remains clear.

The plot in A.2a reveals that two clusters can capture the essential information, as the descent rate of the BIC significantly decreases beyond this point. This observation supports the "elbow method," a heuristic for selecting the optimal number of clusters by identifying where the benefits of additional clusters diminish. By balancing reductions in log-likelihood against an increase in parameters, the method mitigates overfitting risks.

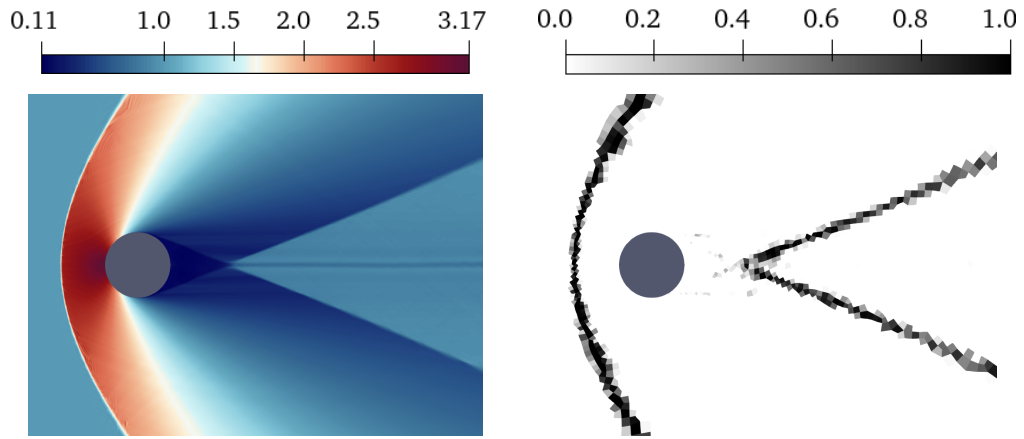
A.3. Comparison with Traditional Shock Sensors

Shock detection is a crucial component of high-fidelity numerical simulations in computational fluid dynamics. Traditional shock sensors, such as the modal sensor of Persson and Peraire [89], and integral sensors, have been widely used to detect discontinuities. These methods rely on local flow properties to infer the presence of shocks but often require parameter tuning and may struggle with complex flow features. The Gaussian Mixture Model (GMM)-based shock detection offers an alternative approach based on unsupervised learning, which automatically clusters flow features into distinct categories without user-defined thresholds.

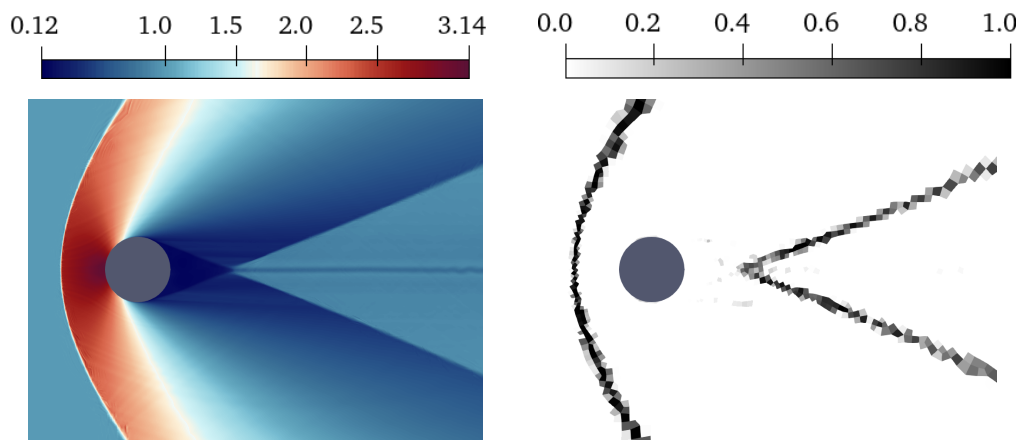
A.3.1. Comparison of GMM with Traditional Shock Sensors

Traditional shock sensors analyze flow properties such as pressure gradients or modal coefficients. The modal sensor, for example, determines shock presence based on the relative weight of high-frequency modes in the polynomial approximation. It effectively captures sharp discontinuities but often requires careful calibration to avoid misidentifying smooth regions as shocks [89]. On the other hand, the integral sensor detects shocks by computing the integral of a specific variable over each computational element. While it provides smooth detection, it is less sensitive to subtle discontinuities, especially in viscous flows. In contrast, the GMM-based sensor automatically adapts to the characteristics of the flow field, distinguishing between shock-dominated and smooth regions based on statistical clustering. Figures A.3, A.4 and A.5 in [73] compare the shock detection results of the modal, integral, and GMM sensors applied to both inviscid and viscous cases. Unlike the modal and integral sensors, the GMM sensor does not suffer from sensitivity to predefined thresholds and can distinguish shocks from boundary layers more effectively. This is particularly relevant in the viscous test case, where traditional sensors may incorrectly classify regions with large velocity gradients as shocks.

The comparison of GMM-based shock detection with traditional methods was evaluated in terms of computational cost and accuracy. Table A.2 presents the relative computational cost of different sensors in the case of the viscous cylinder flow simulation. The GMM sensor, while providing robust shock detection, exhibits a higher computational cost than other methods such as the modal and integral sensors.

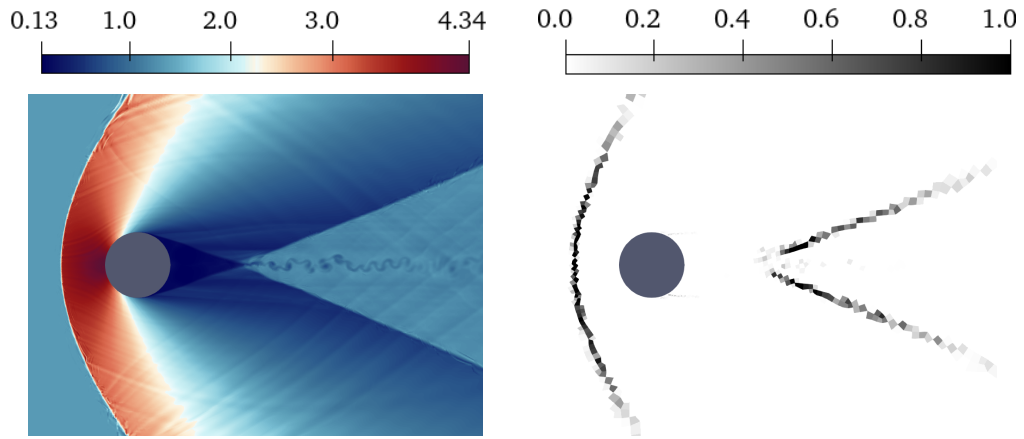


(a) p with $s_0 = -3.5$ and $\Delta s = 1$.

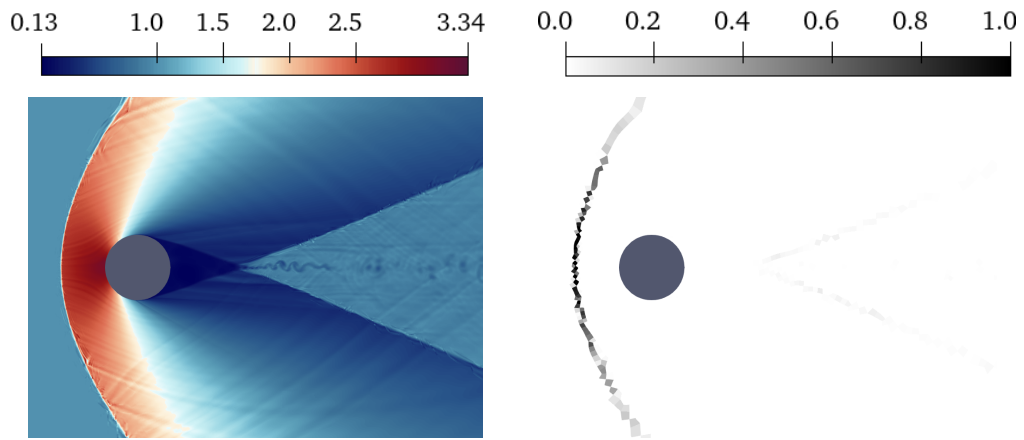


(b) ρ with $s_0 = -3.5$ and $\Delta s = 1$.

Figure A.3.: Viscous case after 300,000 iterations with the modal sensor.



(a) $(\nabla \cdot v)^2$ with $s_0 = 125$ and $\Delta s = 75$.



(b) $\nabla \rho \cdot n_v$ with $s_0 = 15.05$ and $\Delta s = 14.95$.

Figure A.4.: Viscous case after 300,000 iterations with the integral sensor.

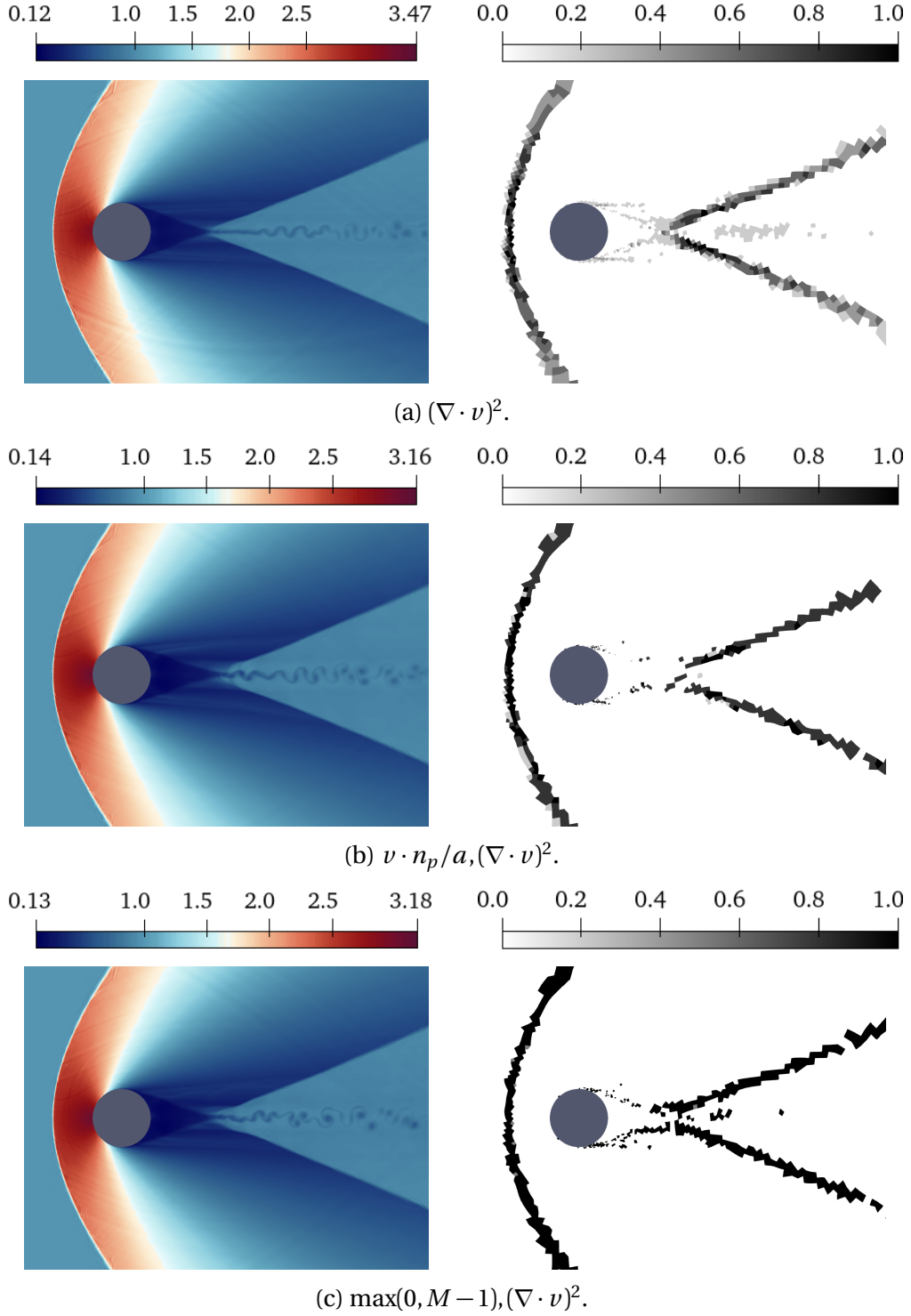


Figure A.5.: Viscous case after 300,000 iterations with our adaptive GMM sensor using 6 clusters. The simulation with the sensor based on $(\nabla \cdot v)^2$ was restarted from the 10,000th time step of the case using $\|\nabla p\|^2, (\nabla \cdot v)^2$.

Table A.2.: Comparison of computational cost for different shock sensors in the viscous cylinder flow simulation. Values represent the percentage of total computational resources used.

Shock Sensor	1 Thread (%)	2 Threads (%)	4 Threads (%)
GMM	12.68	14.16	17.02
Modal	0.71	0.65	0.58
Integral	0.61	0.56	0.53

This analysis indicates that while GMM provides an automatic and adaptive approach to shock detection, it requires significantly higher computational resources. The modal and integral sensors, on the other hand, are computationally cheaper but require predefined thresholds for shock identification.

A.4. Conclusion

This appendix presented a GMM-based shock sensor and a comparison with traditional approaches, including modal and integral sensors. The results demonstrate that while traditional sensors offer lower computational cost, they struggle to distinguish accurately between shocks and other flow features, particularly in viscous cases. In contrast, the GMM-based approach provides a more precise and adaptive classification of flow regions, although at a higher computational cost.

The quantitative comparison shows that GMM achieves the best balance between accuracy and adaptability, making it a promising alternative for shock capturing. However, its increased computational expense suggests that further optimization may be necessary for practical large-scale simulations.

Bibliography

- [1] Introduction To Supervised Machine Learning - The Click Reader.
- [2] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018.
- [3] M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1-2):1–88, 1997.
- [4] I. Babuška and M. Suri. The p and h - p versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632, 1994.
- [5] F. Basile, J.-B. Chapelier, M. de la Llave Plata, R. Laraufie, and P. Frey. Unstructured h-and hp-adaptive strategies for discontinuous Galerkin methods based on a posteriori error estimation for compressible flows. *Computers & Fluids*, 233:105245, 2022.
- [6] A. Bayón. Navier-Stokes Equations | Introduction to CFD. <https://cfd.blogs.upv.es/introduction/navier-stokes-equations/> | Accessed: 15-08-2021 15:00:18.
- [7] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [8] S. Blaise and A. St-Cyr. A dynamic hp-adaptive discontinuous galerkin method for shallow-water flows on the sphere with application to a global tsunami simulation. *Monthly Weather Review*, 140(3):978–996, 2012.
- [9] J. Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.
- [10] M. Braza, P. Chassaing, and H. H. Minh. Numerical study and physical analysis

- of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of fluid mechanics*, 165:79–130, 1986.
- [11] M. Breuer. Large eddy simulation of the subcritical flow past a circular cylinder: numerical and modeling aspects. *International journal for numerical methods in fluids*, 28(9):1281–1302, 1998.
 - [12] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
 - [13] N. Burgess and D. Mavriplis. An hp-adaptive discontinuous Galerkin solver for aerodynamic flows on mixed-element meshes. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 490, 2011.
 - [14] J. S. Cagnone and S. K. Nadarajah. A stable interface element scheme for the p-adaptive lifting collocation penalty formulation. *Journal of Computational Physics*, 231(4):1615–1634, 2012.
 - [15] D. Calhoun. A cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of computational physics*, 176(2):231–275, 2002.
 - [16] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods: Fundamentals in Single Domains*. Springer Science & Business Media, Sept. 2007.
 - [17] G. Chen, Z. Zhang, Z. Tian, L. Li, X. Dong, and H. Ju. Design of a CFD scheme using multiple RANS models for PWR. *Annals of Nuclear Energy*, 102:349–358, Apr. 2017.
 - [18] J.-I. Choi, R. C. Oberoi, J. R. Edwards, and J. A. Rosati. An immersed boundary method for complex incompressible flows. *Journal of Computational Physics*, 224(2):757–784, 2007.
 - [19] J. Christopher, R. D. Falgout, J. B. Schroder, S. M. Guzik, and X. Gao. A space-time parallel algorithm with adaptive mesh refinement for computational fluid dynamics. *Computing and Visualization in Science*, 23(1):13, 2020.
 - [20] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

- [21] B. Cockburn, A. Quarteroni, and Centro internazionale matematico estivo, editors. *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) Held in Cetraro, Italy, June 23-28, 1997*. Number 1697 in Lecture Notes in Mathematics. Springer, Berlin ; New York, 1998.
- [22] C. B. da Silva and J. C. F. Pereira. Invariants of the velocity-gradient, rate-of-strain, and rate-of-rotation tensors across the turbulent/nonturbulent interface in jets. *Physics of Fluids*, 20(5):055101, 2008.
- [23] M. de la Llave Plata, F. Naddei, and V. Couaillier. Les of the flow past a circular cylinder using a multiscale discontinuous Galerkin method. In *5th International conference on Turbulence and Interactions, TI 2018*, 2018.
- [24] J. Dean. *Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners*. John Wiley & Sons, Inc., Hoboken, NJ, USA, Apr. 2014.
- [25] S. Dennis and G.-Z. Chang. Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100. *Journal of Fluid Mechanics*, 42(3):471–489, 1970.
- [26] R. P. Dwight. Goal-oriented mesh adaptation for finite volume methods using a dissipation-based error indicator. *International journal for numerical methods in fluids*, 56(8):1193–1200, 2008.
- [27] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159. Springer, 2004.
- [28] E. Ferrer. *A High Order Discontinuous Galerkin - Fourier Incompressible 3D Navier-Stokes Solver with Rotating Sliding Meshes for Simulating Cross-Flow Turbines*. PhD thesis, Oxford University, UK, 2012.
- [29] E. Ferrer, G. Rubio, G. Ntoulas, W. Laskowski, O. Mariño, S. Colombo, A. Mateo-Gabín, H. Marbona, F. M. de Lara, D. Huergo, et al. Horses3D: A high-order discontinuous Galerkin solver for flow simulations and multi-physics applications. *Computer Physics Communications*, 287:108700, 2023.
- [30] E. Ferrer, G. Rubio, G. Ntoulas, W. Laskowski, O. A. Mariño, S. Colombo, A. Mateo-Gabín, H. Marbona, F. M. de Lara, D. Huergo, et al. : A high-order discontinuous

- galerkin solver for flow simulations and multi-physics applications. *Computer Physics Communications*, 287:108700, 2023.
- [31] J. H. Ferziger and M. Perić. *Computational methods for fluid dynamics*. Springer, 2002.
 - [32] K. J. Fidkowski. Output error estimation strategies for discontinuous galerkin discretizations of unsteady convection-dominated flows. *International Journal for Numerical Methods in Engineering*, 88(12):1297–1322, 2011.
 - [33] K. J. Fidkowski and G. Chen. Metric-based, goal-oriented mesh adaptation using machine learning. *Journal of Computational Physics*, 426:109957, Feb. 2021.
 - [34] B. Fornberg. A numerical study of steady viscous flow past a circular cylinder. *Journal of Fluid Mechanics*, 98(4):819–855, 1980.
 - [35] J. Franke and W. Frank. Large eddy simulation of the flow past a circular cylinder at $Re_D = 3900$. *Journal of wind engineering and industrial aerodynamics*, 90(10):1191–1206, 2002.
 - [36] P. Frey and P.-L. George. *Mesh Generation: Application to Finite Elements*. Hermes Science Publishers, Oxford and Paris, 2000.
 - [37] R. Gandhi. Support Vector Machine — Introduction to Machine Learning Algorithms. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>, July 2018.
 - [38] A. Garai, L. T. Diosady, S. M. Murman, and N. K. Madavan. Dns of low-pressure turbine cascade flows with elevated inflow turbulence using a discontinuous-galerkin spectral-element method. In *Turbo Expo: Power for Land, Sea, and Air*, volume 49712, page V02CT39A025. American Society of Mechanical Engineers, 2016.
 - [39] P. George. Automatic mesh generation and finite element computation. *Handbook of numerical analysis*, 4:69–190, 1996.
 - [40] B. B. . B. Greenwell. *Chapter 19 Autoencoders | Hands-On Machine Learning with R*.
 - [41] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order

- accurate essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, 1987.
- [42] C. Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.
 - [43] J. Hoffman and C. Johnson. Adaptive finite element methods for incompressible fluid flow. In *Error estimation and adaptive discretization methods in computational fluid dynamics*, pages 97–157. Springer, 2003.
 - [44] P. Houston and E. Süli. A note on the design of hp-adaptive finite element methods for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(2):229–243, Feb. 2005.
 - [45] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin. Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.*, 69(12):14413–14423, Dec. 2020.
 - [46] J. Hu, B. K. Ray, and M. Singh. Statistical methods for automated generation of service engagement staffing plans. *IBM Journal of Research and Development*, 51(3.4):281–293, 2007.
 - [47] J. C. Hunt, A. A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. *Studying turbulence using numerical simulation databases, 2. Proceedings of the 1988 summer program*, 1988.
 - [48] J. K. Hunter. *An Introduction to the Incompressible Euler Equations*, 2006.
 - [49] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).
 - [50] K. Jensen. Computational fluid dynamics: Past, present and future. *International Journal of Computational Fluid Dynamics*, 24(3-4):135–145, 2010.
 - [51] D. John and J. Anderson. Computational fluid dynamics: the basics with applications. *Mechanical engineering series*, pages 261–262, 1995.
 - [52] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.
 - [53] S. Kamkar, A. M. Wissink, V. Sankaran, and A. Jameson. Feature-driven cartesian

adaptive mesh refinement for vortex-dominated flows. *Journal of computational physics*, 230(16):6271–6298, 2011.

- [54] P. Knupp and S. Steinberg. *Fundamentals of grid generation*. CRC press, 2020.
- [55] M. Kompenhans, G. Rubio, E. Ferrer, and E. Valero. Adaptation strategies for high order discontinuous Galerkin methods based on tau-estimation. *Journal of Computational Physics*, 306:216–236, 2016.
- [56] M. Kompenhans, G. Rubio, E. Ferrer, and E. Valero. Comparisons of p-adaptation strategies based on truncation- and discretisation-errors for high order discontinuous Galerkin methods. *Computers & Fluids*, 139:36–46, Nov. 2016.
- [57] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Springer Science & Business Media, May 2009.
- [58] D. A. Kopriva and G. Gassner. On the quadrature and weak form choices in collocation type discontinuous galerkin spectral element methods. *Journal of Scientific Computing*, 44:136–155, 2010.
- [59] J. Kou, S. Joshi, A. H. de Mendoza, K. Puri, C. Hirsch, and E. Ferrer. Immersed boundary method for high-order flux reconstruction based on volume penalization. *Journal of Computational Physics*, 448:110721, 2022.
- [60] A. G. Kravchenko and P. Moin. Numerical studies of flow over a circular cylinder at $Re_D = 3900$. *Physics of fluids*, 12(2):403–417, 2000.
- [61] P.-Å. Krogstad and P. E. Eriksen. “blind test” calculations of the performance and wake development for a model wind turbine. *Renewable energy*, 50:325–333, 2013.
- [62] Y.-L. Kuo. The Enhanced r -Refinement Finite Element Analysis of Two-Dimensional Elastic Problems. *Journal of Computational and Theoretical Nanoscience*, 9:908–917, July 2012.
- [63] W. Laskowski, G. Rubio, E. Valero, and E. Ferrer. A functional oriented truncation error adaptation method. *J. Comput. Phys.*, 451:110883, 2022.
- [64] B. Li, Z. Yang, X. Zhang, G. He, B.-Q. Deng, and L. Shen. Using machine learning

to detect the turbulent region in flow past a circular cylinder. *Journal of Fluid Mechanics*, 905, Dec. 2020.

- [65] K. F. Liaw. *Simulation of Flow around Bluff Bodies and Bridge Deck Sections Using CFD*. PhD thesis, University of Nottingham Nottingham, 2005.
- [66] Z. C. Lipton, J. Berkowitz, and C. Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv:1506.00019 [cs]*, Oct. 2015.
- [67] C. Liu, Y. Wang, Y. Yang, and Z. Duan. New omega vortex identification method. *Science China Physics, Mechanics & Astronomy*, 59:1–9, 2016.
- [68] C. Liu, X. Zheng, and C. Sung. Preconditioned multigrid methods for unsteady incompressible flows. *Journal of Computational physics*, 139(1):35–57, 1998.
- [69] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.
- [70] L. Lourenco. Characteristics of the plate turbulent near wake of a circular cylinder. a particle image velocimetry study. *In Unpublished, results taken from Beaudan and Moin*, 1994.
- [71] X. Ma, G.-S. Karamanos, and G. Karniadakis. Dynamics and low-dimensionality of a turbulent near wake. *Journal of fluid mechanics*, 410:29–65, 2000.
- [72] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [73] A. Mateo-Gabín, K. Tlaes, E. Valero, E. Ferrer, and G. Rubio. An unsupervised machine-learning-based shock sensor: Application to high-order supersonic flow solvers. *Expert Systems with Applications*, page 126352, 2025.
- [74] D. J. Mavriplis. Unstructured mesh generation and adaptivity. Technical report, 1995.
- [75] D. S. Mazmuder. *Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods*. Academic Press, Dec. 2015.
- [76] W. F. Mitchell and M. A. McClain. A Comparison of *hp*-Adaptive Strategies for Elliptic Partial Differential Equations. *ACM Trans. Math. Softw.*, 41(1):2:1–2:39, Oct. 2014.

- [77] F. Naddei, M. de la Llave Plata, V. Couaillier, and F. Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods. *Journal of Computational Physics*, 376:508–533, Jan. 2019.
- [78] F. Naddei, M. de la Llave Plata, V. Couaillier, and F. Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous galerkin methods. *Journal of Computational Physics*, 376:508–533, 2019.
- [79] C. Nicholson. Recurrent Networks. <http://wiki.pathmind.com/recurrent-network-rnn>.
- [80] C. Norberg. Effects of Reynolds number and low-intensity free stream turbulence on the flow around a circular cylinder. *Chalmers University of Technology, Department of Applied Thermoscience and Fluid Mechanics, Gothenburg, Sweden, Pub*, 2(87), 1987.
- [81] G. Ntoukas, G. Rubio, O. Marino, A. Liosi, F. Bottone, J. Hoessler, and E. Ferrer. A comparative study of explicit and implicit large eddy simulations using a high-order discontinuous galerkin solver: application to a formula 1 front wing. *Results in Engineering*, 25:104425, 2025.
- [82] J. T. Oden. *Finite elements of nonlinear continua*. Courier Corporation, 2006.
- [83] K.-E. Otmani, A. Mateo-Gabín, G. Rubio, and E. Ferrer. Accelerating high order discontinuous galerkin solvers through a clustering-based viscous/turbulent-inviscid domain decomposition. *Engineering with Computers*, pages 1–16, 2024.
- [84] K.-E. Otmani, G. Ntoukas, O. A. Mariño, and E. Ferrer. Toward a robust detection of viscous and turbulent flow regions using unsupervised machine learning. *Physics of Fluids*, 35(2):027112, 2023.
- [85] H. Ouvrard, B. Koobus, A. Dervieux, and M. V. Salvetti. Classical and variational multiscale LES of the flow around a circular cylinder on unstructured grids. *Computers & Fluids*, 39(7):1083–1094, 2010.
- [86] C. Pang, H. Yang, Z. Gao, and S. Chen. Enhanced adaptive mesh refinement method using advanced vortex identification sensors in wake flow. *Aerospace Science and Technology*, 115:106796, 2021.

- [87] P. Parnaudeau, J. Carlier, D. Heitz, and E. Lamballais. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. *Physics of Fluids*, 20(8):085101, 2008.
- [88] E. Patel and D. S. Kushwaha. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia computer science*, 171:158–167, 2020.
- [89] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous galerkin methods. In *44th AIAA aerospace sciences meeting and exhibit*, page 112, 2006.
- [90] R. M. Pizatella, F. Stanco, and C. Santaera. ENO/WENO INTERPOLATION METHODS FOR ZOOMING OF DIGITAL IMAGES. In *Applied and Industrial Mathematics in Italy II*, pages 480–491, Baia Samuele (Ragusa), Italy, Aug. 2007. WORLD SCIENTIFIC.
- [91] M. Pingjian and Z. Wenping. Numerical simulation of low Reynolds number fluid-structure interaction with immersed boundary method. *Chinese Journal of Aeronautics*, 22(5):480–485, 2009.
- [92] S. Pirozzoli. Generalized conservative approximations of split convective derivative operators. *Journal of Computational Physics*, 229(19):7180–7190, 2010.
- [93] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [94] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.
- [95] L. Rokach and O. Maimon. Clustering methods. *Data mining and knowledge discovery handbook*, pages 321–352, 2005.
- [96] C. Roy. Review of Discretization Error Estimators in Scientific Computing. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2010.
- [97] A. M. Rueda-Ramírez, J. Manzanero, E. Ferrer, G. Rubio, and E. Valero. A p-multigrid strategy with anisotropic p-adaptation based on truncation errors for

high-order discontinuous galerkin methods. *Journal of Computational Physics*, 378:209–233, 2019.

- [98] A. M. Rueda-Ramírez, G. Ntoukas, G. Rubio, E. Valero, and E. Ferrer. Truncation error-based anisotropic p-adaptation for unsteady flows for high-order discontinuous galerkin methods. *International Journal of Computational Fluid Dynamics*, 37(6):430–450, 2023.
- [99] A. M. Rueda-Ramírez, G. Rubio, E. Ferrer, and E. Valero. Truncation error estimation in the p-anisotropic discontinuous galerkin spectral element method. *Journal of Scientific Computing*, 78(1):433–466, 2019.
- [100] R. Russell. Machine Learning: Step-by-Step Guide To Implement Machine Learning Algorithms with Python. page 103, 2018.
- [101] M. Saber, E. R. Abdessamad, S. Rachid, H. Kharraz Aroussi, and M. Wahbi. Artificial Neural Networks, Support Vector Machine and Energy Detection for Spectrum Sensing based on Real Signals. *International Journal of Communication Networks and Information Security*, 11, Apr. 2019.
- [102] E. Saetta and R. Tognaccini. Identification of flow field regions by Machine Learning. In *AIAA SCITECH Forum*. American Institute of Aeronautics and Astronautics, 2022.
- [103] L. Scrucca, M. Saqr, S. López-Pernas, and K. Murphy. An introduction and tutorial to model-based clustering in education via gaussian mixture modelling. *arXiv preprint arXiv:2306.06219*, 2023.
- [104] D. C. Seal. DISCONTINUOUS GALERKIN METHODS FOR VLASOV MODELS OF PLASMA. page 153, 2012.
- [105] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.
- [106] C.-W. Shu, T. A. Zang, G. Erlebacher, D. Whitaker, and S. Osher. High-order ENO schemes applied to two- and three-dimensional compressible flow. *Applied Numerical Mathematics*, 9(1):45–71, 1992.

- [107] C.-W. Shu and Y. Zeng. High-order essentially non-oscillatory scheme for viscoelasticity with fading memory. *Quart. Appl. Math.*, 55(3):459–484, Sept. 1997.
- [108] J. P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis. Cfd vision 2030 study: a path to revolutionary computational aerosciences. Technical report, 2014.
- [109] J. Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.
- [110] D. O. Snyder and G. Degrez. Large-eddy simulation with complex 2-D geometries using a parallel finite-element/spectral algorithm. *International journal for numerical methods in fluids*, 41(10):1119–1135, 2003.
- [111] J. N. Sorensen and W. Z. Shen. Numerical modeling of wind turbine wakes. *J. Fluids Eng.*, 124(2):393–399, 2002.
- [112] S. Sunthornjittanon. Linear Regression Analysis on Net Income of an Agrochemical Company in Thailand. Technical report, University Honors College, Jan. 2015.
- [113] O. Theobald. Machine Learning For Absolute Beginners. *O'Reilly Media theobald-MachineLearningAbsolute*, page 128, 2017.
- [114] J. F. Thompson. Grid generation techniques in computational fluid dynamics. *AIAA journal*, 22(11):1505–1523, 1984.
- [115] J. F. Thompson, Z. U. Warsi, and C. W. Mastin. *Numerical grid generation: foundations and applications*. Elsevier North-Holland, Inc., 1985.
- [116] J. Tinsley. An hp adaptive strategy for finite element approximation of the navier-stokes equations.
- [117] N. Ueda and R. Nakano. Mixture density estimation via em algorithm with deterministic annealing. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 69–77. IEEE, 1994.
- [118] H. K. Versteeg. *An introduction to computational fluid dynamics the finite volume method, 2/E*. Pearson Education India, 2007.
- [119] A. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of fluids*, 16(10):3670–3681, 2004.

- [120] J. G. Wallwork, J. Lu, M. Zhang, and M. D. Piggott. E2n: Error estimation networks for goal-oriented mesh adaptation. *arXiv preprint arXiv:2207.11233*, 2022.
- [121] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al. High-order cfd methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.
- [122] M. Webster. Merriam-webster online dictionary. 2009. URL <http://www.merriamwebster.com>, 2009.
- [123] J. Wells, A.-H. Salem Said, and S. Ragab. Effects of Turbulence Modeling on RANS Simulations of Tip Vortices. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, Florida, Jan. 2010. American Institute of Aeronautics and Astronautics.
- [124] J. Williamson. Low-storage Runge-Kutta schemes. *Journal of Computational Physics*, 35(1):48–56, 1980.
- [125] F. Yu and L. T. Zhang. Stabilized finite element formulation and high-performance solver for slightly compressible navier–stokes equations. In *Advances in Fluid Mechanics: Modelling and Simulations*, pages 209–232. Springer, 2022.
- [126] Y. Zhou. Implications of turbulence interactions: A path toward addressing very high Reynolds number flows. Technical Report UCRL-CONF-221402, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), May 2006.
- [127] Y. Zhou, K. Nagata, Y. Sakai, Y. Ito, and T. Hayase. On the evolution of the invariants of the velocity gradient tensor in single-square-grid-generated turbulence. *Physics of Fluids*, 27:075107, 2015.
- [128] H. Zhu, H. Wang, and Z. Gao. A new troubled-cell indicator for discontinuous galerkin methods using k-means clustering. *SIAM Journal on Scientific Computing*, 43(4):A3009–A3031, 2021.
- [129] X. Zhu and A. B. Goldberg. Introduction to Semi-Supervised Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, Jan. 2009.
- [130] M. Zormpa, S. Le Clainche, E. Ferrer, C. Vogel, and R. Willden. Dynamic mode

decomposition of merging wind turbine wakes. In *Journal of Physics: Conference Series*, volume 2505, page 012020. IOP Publishing, 2023.