Numerically-informed neural networks for degree adaptive unsteady incompressible flow simulations

Swansea University
Prifysgol Abertawe

Agustina Felipe

College of Engineering

Swansea University

This dissertation is submitted for the degree of

Doctor of Philosophy

January 2025

Declaration

Statement 1

No part of this work has previously been submitted for any degree and is not being concurrently submitted in candidature for any degree at this or any other university.

Date: 24/01/2025

Signature: ....[redacted]..... (Candidate)

Statement 2

This thesis is the result of my own investigations, references to the work of others have been clearly acknowledged. A bibliography is appended.

Date: 24/01/2025

Signature: ....[redacted]..... (Candidate)

Statement 3

I hereby give consent for my thesis, if accepted, to be available for electronic sharing **after expiry of a bar on access approved by the Swansea University.**

Date: 24/01/2025

Signature: ....[redacted]..... (Candidate)

Statement 4

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Date: 24/01/2025

Signature: ....[redacted]..... (Candidate)

Abstract

The need for transient incompressible flow simulations in science and engineering has driven the demand for high-order methods over conventional low-order finite element and finite volume approaches. High-order methods offer greater accuracy and efficiency in capturing the complex, time-dependent behaviour of fluid systems because of the lower dissipation and dispersion of high-order approximations. Traditional low-order methods often require highly refined meshes to achieve comparable accuracy, leading to higher computational costs.

This thesis focuses on problems where flow features such as vortices or gust perturbations need to be propagated over long distances. These flow features can be more accurately propagated using high-order methods, but their localised nature suggests that incorporating degree adaptive schemes can lead to significantly more efficient simulations by only employing high-order approximations where needed. Discontinuous Galerkin methods have gained significant popularity and provide an easy-to-implement framework for degree adaptivity. In particular, the hybridisable discontinuous Galerkin is adopted in this work and implemented in Fortran 90.

This thesis provides two original scientific contributions. First, a conservative projection scheme has been developed and implemented to enable efficient degree adaptive simulations for transient incompressible flows. The proposed scheme is found to remove all the numerical artefacts shown by a standard adaptive process due to the violation of the free-divergence condition when projecting a solution from a space of polynomials of a given degree to a space of polynomials with a lower degree. Second, a novel degree adaptive procedure is designed by using a trained artificial neural network to predict the solution at a future time from the solution at the current time. The procedure is shown to perform the degree adaptivity in places where flow features will travel in the future and prevents the traditional requirement to perform degree adaptivity cycles within a time step.

Keywords: transient incompressible flow, high-order methods, hybridisable discontinuous Galerkin, degree adaptivity, artificial neural network

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Rubén and Oubay, for their invaluable guidance, unwavering support, and profound expertise throughout this research journey.

My heartfelt appreciation extends to my family: my parents, whose endless love and encouragement have been my foundation; my siblings Santiago, Cecilia and Federico, who have always believed in me; and my nephew Valentino, whose joy brings light to my life.

Finally, I wish to thank all my friends who have supported me throughout this endeavour. Your friendship has been an invaluable source of strength and motivation.

Nomenclature

Latin Symbols

b	Aggressiveness parameter (base for logarithm in degree adaptation)
$\mathbf{B}^X$	Set of buckets for input variable X
$\mathbf{B}^Y$	Set of buckets for output variable Y
d	Stencil distance
E	Error indicator
h	Characteristic element size
k	Polynomial degree of approximation
$\boldsymbol{L}$	Exact velocity gradient tensor
$\mathbf{L}$	Approximate velocity gradient tensor
$\boldsymbol{n}$	Outward unit normal vector
$\mathtt{n}_{\mathtt{adaptivity}}$	Number of iterations of the adaptive process
$\mathtt{n}_{\mathtt{buckets}}$	Number of buckets for data reduction
$\mathtt{n}_{\mathtt{cases}}$	Number of training cases
$\mathtt{n}_{\mathtt{el}}$	Number of elements
$\mathtt{n}_{\mathtt{en}}$	Number of element nodes
$\mathtt{n}_{\mathtt{epochs}}$	Number of training epochs
$\mathtt{n}_{\mathtt{fn}}$	Number of face nodes
$\mathtt{n}_{\mathtt{param}}$	Number of flow parameters
$\mathtt{n}_{\mathtt{sd}}$	Number of spatial dimensions

$\mathtt{n_{stencil}}$	Number of points in stencil
$\mathtt{m_{sd}}$	Number of components in symmetric tensor (3 in 2D, 6 in 3D)
p	Exact pressure field
p_h	Approximate pressure field
$\boldsymbol{s}$	Volumetric source term
t	Time
T	Final time
$\boldsymbol{t}$	Prescribed traction on Neumann boundary
$\boldsymbol{u}$	Exact velocity field
$\mathbf{u}$	Approximate velocity field
$\hat{\mathbf{u}}$	Trace of approximate velocity field on mesh skeleton
$\boldsymbol{x}$	Spatial coordinates
$\mathbf{X}$	Input data matrix
$\mathbf{Y}$	Output data matrix

Greek Symbols

Γ	Domain boundary
Γ_D	Dirichlet boundary
Γ_N	Neumann boundary
ε	Target error tolerance
λ	Lagrange multiplier
ν	Kinematic viscosity ($= 1/\mathrm{Re}$)
Ω	Computational domain
Ω_e	Element domain
ρ	Mean pressure on element boundaries
τ	Tangential direction vector
τ_a	Convective stabilization parameter
τ_d	Diffusive stabilization parameter

Dimensionless Numbers

Re	Reynolds number

Spaces

$\mathcal{G}_h$	Space of polynomial functions for velocity gradient tensor
$\mathcal{V}_h$	Space of polynomial functions for velocity field
$\mathcal{P}_h$	Space of polynomial functions for pressure
$\mathcal{M}_h$	Space of polynomial functions for trace velocity
$\mathcal{W}_h$	Space of constant functions on element boundaries
$L^2(\Omega)$	Space of square-integrable functions
$H^1(\Omega)$	Space of functions with square-integrable gradients
$H(\mathrm{div};\Omega)$	Space of vector functions with square-integrable divergence

Operators and Notation

$(\cdot,\cdot)_{\mathcal{T}_h}$	Inner product over computational domain
$\langle\cdot,\cdot\rangle_{\partial\mathcal{T}_h}$	Inner product over mesh skeleton
$[\![\cdot]\!]$	Jump operator across interior faces
∇	Gradient operator
$\nabla\cdot$	Divergence operator
$\nabla^{\mathtt{s}}$	Symmetric gradient operator

Subscripts and Superscripts

$(\cdot)_h$	Discrete/approximate quantity
$(\cdot)_e$	Elemental quantity

$(\cdot)^*$	Post-processed quantity
$(\cdot)_D$	Related to Dirichlet boundary
$(\cdot)_N$	Related to Neumann boundary
$(\hat{\cdot})$	Quantity defined on mesh skeleton

Note: All variables are presented in their non-dimensional form following appropriate scaling of the governing equations.

Table of contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The simulation of incompressible transient flows, where transient refers to time-dependent or unsteady phenomena that vary with time, plays a key role in both engineering and research, helping to understand and predict a wide variety of natural and industrial events in areas such as aerodynamics, weather forecasting, cardiovascular circulation, and industrial blending.

Incompressible transient flow simulations present several challenges, both mathematically and numerically. One of the primary difficulties arises from the incompressibility condition, which requires that the divergence of the velocity field remains zero at all times. This introduces a strong coupling between the pressure and velocity fields, making their resolution difficult. Additionally, because information in an incompressible medium theoretically propagates at infinite speed, it places stringent demands on numerical methods to avoid unphysical results. This can lead to issues with stability and accuracy, particularly in time-dependent simulations, where the temporal evolution of the flow must be accurately captured. Ensuring incompressibility at each time step often requires special techniques such as projection methods or pressure correction schemes, which can add computational complexity (Donea and Huerta, 2003).

The vast majority of industrial, commercial, and academic flow solvers adopt low-order approaches based on finite differences, finite element, or finite volume methods (Deng et al., 1996; Franca and Frey, 1992; Li et al., 2022; Whiting and Jansen, 2001). Finite difference methods in Cartesian grids are known to be extremely efficient but are unable to handle complex geometries without the time-consuming human intervention required to perform a block subdivision of the geometric model (Ali et al., 2017). Finite element and finite volume methods enable the use of unstructured meshes that provide the preferred framework to simulate problems involving complex geometries without the need of human intervention to generate meshes and enabling local mesh refinement. These methods have proven to be extremely robust and competitive for the simulation of steady flows (Morgan et al., 1991; Gerhold, 2005; Biedron et al., 2016).

The simulation of transient flows using low order schemes poses significant challenges, mainly due to the high dissipation and dispersion errors associated with low order approximations (Ainsworth et al., 2006). The need to use extremely refined meshes to capture the transient flow features that propagate over long distances seems to suggest that high order methods, where high-order refers to numerical methods that use polynomial approximations of degree greater than one to represent the solution within each element, offering potentially higher accuracy per degree of freedom compared to traditional linear approximations, are better suited for applications involving unsteady phenomena.

In the last two decades, there has been significant interest in developing high-order finite volume and finite element schemes (Nogueira et al., 2010; Campagne et al., 2010; Chalot and Normand, 2010; Sevilla et al., 2013; Chalot et al., 2015). Traditional high-order finite volume methods require the definition of a stencil to perform the high-order approximation. This introduces difficulties when using unstructured meshes and requires some special treatment near boundaries. High-order finite element methods have shown benefits when compared to low order elements, but the definition of the stabilisation, especially for highly stretched elements, seems to be the major difficulty that prevents the adoption of such schemes.

Another class of high-order methods that have gained substantial interest from the research community is discontinuous Galerkin (DG) methods (Cockburn, 2017). Initially developed to solve neutron transport equations, DG methods gained popularity due to the work of Shu and Cockburn published in a series of five articles (Cockburn and Shu, 1991, 1989; Cockburn et al., 1989, 1990; Cockburn and Shu, 1998b).

The first extension of DG methods to convection-diffusion problems was proposed in (Arnold, 1982), where the interior penalty method was introduced using discontinuous elements. The method showed optimal convergence, but only when using a mesh dependent penalty parameter.

The extension of DG methods to handle the elliptic operator (e.g. second order derivatives), required to extend the initial work on hyperbolic systems to solve problems governed by the Navier-Stokes equations, was object of intensive research during the late 1990s (Cockburn and Shu, 1998a; Baumann and Oden, 1999). The local DG method, introduced in (Cockburn and Shu, 1998a), was widely adopted for the solution of convection-diffusion problems and further improved to reduce its stencil, leading to the compact DG method (Peraire and Persson, 2008).

DG methods offer greater flexibility compared to continuous finite elements in handling non-conforming meshes and non-uniform polynomial approximations. While continuous Galerkin methods can also accommodate non-uniform polynomial orders, they require additional constraints and continuity enforcement at element interfaces that increase implementation complexity. The definition of stabilisation for convection-dominated problems is generally more straightforward within a DG framework. Furthermore, in the context of incompressible flows, DG methods enable the use of identical polynomial spaces for velocity and pressure approximations. This is not the case for continuous Galerkin approaches that usually require a lower dimensional space for the pressure compared to the velocity in order to guarantee the Ladyzhenskaya-Babuška-Brezzi (LBB) condition (Donea and Huerta, 2003). The higher-order pressure approximation allows for more accurate representation of pressure gradients, which is

particularly important in problems involving boundary layers or rapid pressure variations. The ability to handle discontinuities at element interfaces through numerical fluxes, combined with intrinsic stabilisation, makes DG methods particularly robust for convection-dominated flows without requiring additional stabilisation parameters or shock-capturing terms that are typically needed in continuous formulations.

The attractive properties of the DG methods led to application to the solution of incompressible viscous flows using the local DG method (Cockburn et al., 2005) and the development of methods to employ solenoidal basis (Montlaur et al., 2010) and DG methods that employed the same order of approximation for velocity and pressure (Cockburn et al., 2009b).

Despite the advantages of DG schemes, the main criticism received by these methods has traditionally been the increased computational cost due to the use of discontinuous approximation spaces, meaning that degrees of freedoms on the inter-element faces are duplicated. The hybridisable DG (HDG) method was introduced by Cockburn and co-workers to alleviate this problem (Cockburn et al., 2009a).

As other DG methods, the HDG method uses a mixed formulation where the primal variable (velocity field for incompressible flows) and its gradient are considered as independent unknowns. The most distinctive feature of the HDG method is the introduction of the so-called *hybrid* variable on the faces of the elements (edges in two dimensions), corresponding to the trace of the velocity on the mesh skeleton. Using the hybridisation process, which is equivalent to the static condensation commonly used in continuous Galerkin approaches (Guyan, 1965), the hybridisation leads to a reduced system of equations where only the degrees of freedom associated to the element faces are globally coupled. An important property of HDG methods, contrary to other DG methods, is the possibility to build a super-convergent approximation of the primal variable. The ability to work with the spaces of polynomials with the same degree for both the primal and mixed variable and the ability to deliver optimal rates of convergence ($k+1$ in the $\mathcal{L}^2(\Omega)$ norm when employing an approximation with polynomials

of degree k) enables the possibility of building a super-convergent, or postprocessed, solution (with a rate of convergence $k+2$).

HDG methods for incompressible flows were developed during the 2010s, primarily focussing on the development of HDG schemes for Stokes flows (Cockburn et al., 2010; Nguyen et al., 2010; Cockburn et al., 2011; Cockburn and Shi, 2013, 2014). The extension to incompressible viscous flows governed by the Navier-Stokes equations was first presented in (Nguyen et al., 2011). Early work on these methods (Nguyen et al., 2010) showed that mixed formulations based on the Cauchy stress tensor exhibited a suboptimal rate of convergence of the mixed variable, which in turn led to a loss of the superconvergent properties of the postprocessed velocity. This phenomenon had also previously been reported for the solution of linear elasticity problems with the HDG method (Soon et al., 2009). In (Sevilla et al., 2018) and (Giacomini et al., 2018), the authors linked this phenomenon to the weak imposition of the symmetry of the stress tensor, for elasticity and Stokes flows, respectively. The authors proposed the use of the so-called Voigt notation, which has been traditionally used by the solid mechanics community and that can be seen as a strong imposition of the symmetry of the stress tensor.

The ability of HDG methods to build a super-convergent approximation of the primal variable (the velocity in incompressible flow problems) was identified in (Giorgiani et al., 2013) as a route to devise a cheap error indicator, based on the difference between the primal and superconvergent solutions. This strategy has been employed in the context of Stokes and Navier-Stokes flows (Giorgiani et al., 2014; Sevilla and Huerta, 2018).

1.2 Objectives

The problems of interest in this work involve the propagation of local flow features such as vortices or gust perturbations over long distances. In a low-order context, efficient

simulations that involve the propagation of local flow features require the use of mesh adaptivity. However, in a high-order context, the possibility to locally adapt the degree of approximation without changing the mesh topology is an attractive alternative. The use of DG methods in general enables an easy framework to handle different degrees of approximation in different elements. Furthermore, the HDG method enables us to devise a cheap error indicator to drive the adaptive process.

The main challenge is the ability to design efficient robust and efficient degree adaptive approaches for simulating transient viscous incompressible flows. To this end, this thesis aims at solving two issues that are often encountered when applying degree adaptive schemes to the solution of the unsteady incompressible Navier-Stokes equations.

Degree adaptive schemes often assume that the degree of approximation is not lowered during a simulation. The main issue of lowering the degree of approximation when simulating incompressible flows is the potential violation of the free-divergence condition. This issue is obviously not relevant when performing steady simulations because the result of one simulation is only used as an initial guess of the iterative method employed to solve the resulting non-linear system of equations after discretising the problem in space and time. However, in the context of transient flows, lowering the degree of approximation leads to a velocity field that is not divergent free and it is used to advance the solution to the next time step. As a result, the accuracy and robustness of the method can be lost. The problem does not appear when the degree of approximation is increased because the space of polynomials of a certain degree is contained in the space of polynomials of a higher degree. Therefore, the projection of the solution from a space of polynomials of a given degree into a space of higher order polynomials do not modify the approximation.

Despite degree adaptive approaches that do not allow lowering the degree of approximation are robust, they are not efficient. Especially when local flow features travel over long distances, these approaches can lead to an unnecessary high computational cost. Many elements of the mesh might be using high order approximations long after

a particular flow feature passed through that element, even if this accuracy is no longer required.

The first objective is to develop an easy-to-implement and efficient strategy to enable lowering the degree of approximation in a transient degree adaptive process. The key idea is to develop a projection from a space of polynomials to another space of polynomials with a lower degree that guarantees that the projected velocity is divergence-free.

To deliver high accuracy and efficiency, high-order spatial discretisation schemes need to be implemented with high-order time integrators. In this work, high order explicit singly diagonal implicit Runge-Kutta (ESDIRK) integration methods are considered. With these methods, large time steps can be employed, allowing flow features to travel distances larger than the element size during a single time step. In this context, traditional adaptive approaches require repetition of the adaptive process within a time step to ensure that the elements have enough resolution to accurately represent high solution gradients in the next time step.

The second objective is to accelerate a degree adaptive process by using a trained artificial neural network to predict the solution at a future time from the solution at the current time. The aim is to eliminate the need to repeat the adaptive process during a time step, substantially accelerating the simulation without loss of accuracy.

1.3 Outline of the thesis

The remaining of the thesis is organised as follows.

Chapter 2 briefly recalls the transient incompressible Navier-Stokes equations and describes the fundamentals of the HDG method. Details about the weak formulation, the spatial discretisation with high-order isoparametric elements and the temporal discretisation with backward differentiation formulae (BDF) and explicit first stage, singly diagonally implicit Runge-Kutta (ESDIRK) methods are presented. Details

about the linearisation procedure using the Newton-Raphson method are given and some numerical examples are introduced to verify the implemented Fortran 90 code in two and three dimensions.

Chapter 3 presents the first original contribution of the thesis, which involves a novel conservative projection for the simulation of transient incompressible flows using degree adaptivity. The strategy to build a super-convergent approximation of the velocity in an HDG context is presented, and its use to devise a cheap error indicator to drive a degree adaptive process is detailed. The proposed conservative projection is described, and a discussion is presented about the need of this strategy in transient problems where the degree of approximation is to be lowered in regions of the domain to maintain efficiency. A numerical example is used to verify the implementation of the degree adaptive process for steady state problems. Finally, a more challenging transient problem is considered to assess the performance and accuracy of the proposed conservative projection. The results are compared to the results obtained with traditional adaptive processes, and also compared to approaches where the degree of approximation is not allowed to be lowered.

Chapter 4 presents the second original contribution of the thesis, which is the development of a novel neural network-driven degree adaptive strategy. After providing a brief review of artifical neural networks, the proposed strategy is described. The main idea is to train a network to predict the solution at a given mesh node at time t^{n+1} from the solution at the same node and at a number of surrounding points at time t^n. The network architecture is detailed, and crucial details are given to perform the data acquisition and preparation before the network is trained. A verification example is introduced in this chapter to illustrate all the steps involved, from data collection, data preparation, training of the network, and deployment within a degree adaptive process.

Chapter 5 focusses on the application of developed technology for the simulation of problems involving the propagation of gust perturbation over long distances. Two examples are presented to illustrate the benefits of the two original contributions described

in Chapters 3 and 4. The first example involves the simulation of a gust impinging on a NACA0012 aerofoil, and the advantages of using the conservative projection within a degree adaptive process are analysed. The second example involves the propagation of a parametric gust in a free-stream flow, and the benefits of using a degree adaptive process driven by a trained neural network are demonstrated. This example also provides a series of numerical experiments to study the performance of different neural network architectures, as well as the influence of the numerical parameters introduced in this approach.

Finally, chapter 6 summarises the conclusions of the work presented and describes some potential avenues for future research.

Chapter 2

The HDG method for transient incompressible flows

2.1 Introduction

The numerical simulation of incompressible and unsteady flows governed by the Navier-Stokes equations remains a fundamental challenge in computational fluid dynamics (CFD). These equations, which describe the motion of viscous fluid substances, are of crucial importance in a wide range of scientific and engineering applications, from weather prediction to aerodynamics (Temam, 2001). However, their inherent non-linearity and the need to satisfy the incompressibility constraint pose significant computational difficulties, particularly for high Reynolds number flows and complex geometries (Quarteroni, 2017).

In recent years, the hybridisable discontinuous Galerkin (HDG) method has emerged as a promising approach to the numerical solution of partial differential equations, including incompressible flows. Cockburn and co-workers devised and analysed various HDG formulations for Stokes flows (Cockburn et al., 2010, 2011; Nguyen et al., 2010). In (Nguyen et al., 2010) the authors observed a sub-optimal convergence of the mixed variable for low-order approximations and consequently a loss of optimal

convergence of the postprocessed velocity. Giacomini et al. (2018) proposed a simple remedy by strongly imposing the symmetry of the stress tensor, in practice using the so-called Voigt notation. An alternative formulation to guarantee optimal convergence of the mixed variable when employing low order approximations, using the so-called M-decompositions, was proposed in (Cockburn and Fu, 2017).

The development of HDG methods for the incompressible Navier-Stokes equations was first presented in (Nguyen et al., 2011) and the mathematical analysis was presented in (Cesmelioglu et al., 2017). The ability to build a super-convergent velocity field was exploited in (Giorgiani et al., 2014) to devise a cheap error indicator to drive a degree adaptive process. For a more comprehensive literature review on HDG methods, including the application to the incompressible Navier-Stokes equations, the reader is referred to the tutorial presented in (Giacomini et al., 2020).

The application of HDG to the incompressible Navier-Stokes equations offers several potential benefits:

- Optimal convergence rates for velocity, pressure, and velocity gradients
- Element-by-element postprocessing yielding superconvergent velocity fields
- Exact satisfaction of the incompressibility constraint at the discrete level
- Reduced number of globally coupled degrees of freedom compared to standard DG methods
- Flexible and robust treatment of boundary conditions

This chapter summarises the HDG formulation of the incompressible unsteady Navier-Stokes equations. Spatial discretisation is based on the HDG framework introduced by Nguyen et al. (2011); Giacomini et al. (2020), whilst temporal discretisation employs two families of implicit methods: backward differentiation formulae (BDF) (Curtiss and Hirschfelder, 1952) and explicit first stage, singly diagonally implicit Runge-Kutta

(ESDIRK) methods (Kennedy and Carpenter, 2003). These temporal discretisation schemes are chosen for their stability properties and ability to handle stiff problems efficiently.

The remainder of this chapter is organised as follows. Sections 2.3 to 2.6 present the mathematical formulation of the HDG method for the unsteady incompressible Navier-Stokes equations, including spatial and temporal discretisations. Section 2.7 discusses the details of the Newton-Raphson linearisation strategy and the hybridisation process. Section 2.8 provides a set of numerical examples to verify the optimal spatial and temporal convergence properties of the implemented numerical scheme.

The numerical methods and algorithms presented in this work have been implemented from scratch in Fortran 90, enabling complete control over all aspects of the implementation. Whilst building on established theoretical foundations, the code base was developed independently to optimise performance and maintain flexibility for future extensions. This ground-up implementation approach allows for detailed verification of the properties of the method and facilitates modifications to explore novel numerical techniques. The custom implementation provides full access to all intermediate quantities needed for analysis and ensures reproducibility of the results presented in subsequent chapters.

2.2 The transient incompressible Navier-Stokes equations

The transient incompressible Navier-Stokes problem in the open bounded computational domain $\Omega \in \mathbb{R}^{\mathtt{n_{sd}}}$ with boundaries $\partial\Omega$ and $\mathtt{n_{sd}}$ denoting the number of spatial

dimensions is expressed as

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \boldsymbol{\nabla} \cdot (2\nu \boldsymbol{\nabla}^{\mathtt{s}} \boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}) = \boldsymbol{s} & \text{in } \Omega \times (0, T] \quad (2.1a) \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 & \text{in } \Omega \times (0, T] \quad (2.1b) \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } \Gamma_D \times (0, T] \quad (2.1c) \\ \boldsymbol{n} \cdot (2\nu \boldsymbol{\nabla}^{\mathtt{s}} \boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}} - \boldsymbol{u} \otimes \boldsymbol{u}) = \boldsymbol{t} & \text{on } \Gamma_N \times (0, T] \quad (2.1d) \\ \boldsymbol{u} = \boldsymbol{u}_0 & \text{in } \Omega \times \{0\} \quad (2.1e) \end{cases}$$

Equation (2.1a) represents the momentum equation, where $\boldsymbol{\nabla}^{\mathtt{s}} := \left(\boldsymbol{\nabla} + \boldsymbol{\nabla}^T\right)/2$ is the symmetric gradient operator, $\boldsymbol{u}$ denotes the velocity field, p is the pressure, ν is the kinematic viscosity, $\mathbf{I}_{\mathtt{n_{sd}}}$ is the identity tensor of dimensions $\mathtt{n_{sd}}$, $\partial \boldsymbol{u}/\partial t$ is the temporal derivative of the velocity field, T is the final time and $\boldsymbol{s}$ represents the body force. The Reynolds number is given by $Re = 1/\nu$.

Equation (2.1b) is the continuity equation, also known as the incompressibility condition, which enforces a divergence-free velocity field.

The boundary $\partial\Omega$ is partitioned into two disjoint parts: the Dirichlet portion Γ_D and the Neumann portion Γ_N, such that $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$. Equation (2.1c) imposes a Dirichlet boundary condition, specifying the velocity $\boldsymbol{u}_D$ on Γ_D. Equation (2.1d) prescribes a Neumann boundary condition for the momentum equation, where $\boldsymbol{t}$ is the traction specified on Γ_N, and $\boldsymbol{n}$ is the normal outward unit vector.

The term $\partial \boldsymbol{u}/\partial t$ accounts for the unsteadiness of the flow, indicating that velocity and pressure evolve with time, and (2.1e) represents the initial condition. Consequently, the solution of these equations requires the application of time integration methods in the time interval $(0, T]$.

It is important to note that the incompressibility condition (2.1b) imposes a compatibility constraint on the velocity field. Applying the divergence theorem to the integral

of (2.1b) leads to

$$\int_{\Omega} \boldsymbol{\nabla} \cdot \boldsymbol{u} \mathrm{d}\Omega = \int_{\partial\Omega} \boldsymbol{u} \cdot \boldsymbol{n} \, \mathrm{d}\Gamma = 0. \tag{2.2}$$

Using the Dirichlet boundary condition, the compatibility constraint can be written as

$$\int_{\Gamma_D} \boldsymbol{u}_D \cdot \boldsymbol{n} \, \mathrm{d}\Gamma + \int_{\Gamma_N} \boldsymbol{u} \cdot \boldsymbol{n} \, \mathrm{d}\Gamma = 0. \tag{2.3}$$

In addition, when exclusively Dirichlet boundary conditions are prescribed (i.e. $\Gamma_D = \partial\Omega$), the pressure is only determined up to a constant. Imposing only Dirichlet boundary conditions is equivalent to prescribing only Neumann boundary conditions on the numerical flux, which leads to an ill-posed problem that requires an additional constraint on the velocity field. Therefore, an additional constraint on the pressure field must be imposed to eliminate its indeterminacy. For hybrid formulations, it is common practice to enforce a zero mean pressure condition on the boundary (see, for example, Cockburn et al. (2009a), Cockburn et al. (2010), Cockburn and Shi (2014)). This condition is expressed as

$$\int_{\partial\Omega} p \, \mathrm{d}\Gamma = 0. \tag{2.4}$$

2.3 HDG strong forms

The computational domain Ω is partitioned into a set of non-overlapping subdomains, or elements, namely

$$\Omega = \bigcup_{e=1}^{\mathtt{n_{el}}} \Omega_e, \quad \text{such that} \quad \Omega_i \cap \Omega_j = \emptyset \quad \text{for} \quad i \neq j. \tag{2.5}$$

The boundaries $\partial\Omega_e$ of these subdomains define an internal interface, also called mesh skeleton, defined as

$$\Gamma := \left[\bigcup_{e=1}^{\mathtt{n_{el}}} \partial\Omega_e \right] \setminus \partial\Omega. \tag{2.6}$$

Finally, the scalar products $(\cdot,\cdot)_D$ and $\langle\cdot,\cdot\rangle_B$ are introduced, denoting, respectively, the $\mathcal{L}^2$ scalar product in any domain $D \subset \Omega$ and the $\mathcal{L}^2$ scalar product of the traces over $B \subset \partial\Omega$.

Following (Sevilla and Huerta, 2016; Giacomini et al., 2020), we introduce the discrete functional spaces

$$\begin{aligned} \mathcal{V}^h(\Omega) &:= \left\{ v \in \mathcal{L}_2(\Omega) : v|_{\Omega_e} \in \mathcal{P}^k(\Omega_e) \ \forall \Omega_e,\ e = 1, \dots, \mathtt{n_{el}} \right\}, \\ \hat{\mathcal{V}}^h(S) &:= \left\{ \hat{v} \in \mathcal{L}_2(S) : \hat{v}|_{\Gamma_i} \in \mathcal{P}^k(\Gamma_i) \ \forall \Gamma_i \subset S \subseteq \Gamma \cup \partial\Omega \right\}, \end{aligned} \tag{2.7}$$

where $\mathcal{P}^k(\Omega_e)$ and $\mathcal{P}^k(\Gamma_i)$ denote the spaces of polynomial functions of complete degree at most k in Ω_e and on Γ_i, respectively.

As commonly done in DG methods, the original system of partial differential equations is written on the so-called broken computational domain by imposing the continuity of the velocity and the fluxes on the mesh skeleton, namely

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot(2\nu\nabla^{\mathtt{s}}\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}) = \boldsymbol{s} & \text{in } \Omega_e \times (0,T] & \text{(2.8a)} \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \times (0,T] & \text{(2.8b)} \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } (\partial\Omega_e \cap \Gamma_D) \times (0,T] & \text{(2.8c)} \\ \boldsymbol{n}\cdot(2\nu\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}} - \boldsymbol{u}\otimes\boldsymbol{u}) = \boldsymbol{t} & \text{on } (\partial\Omega_e \cap \Gamma_N) \times (0,T] & \text{(2.8d)} \\ \boldsymbol{u} = \boldsymbol{u}_0 & \text{in } \Omega_e \times \{0\} & \text{(2.8e)} \\ [\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \mathbf{0} & \text{on } \Gamma \times (0,T] & \text{(2.8f)} \\ [\![((\sqrt{2\nu}\boldsymbol{L} - p\mathbf{I}_{\mathtt{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n}]\!] = \mathbf{0} & \text{on } \Gamma \times (0,T] & \text{(2.8g)} \end{cases}$$

where, following the definition of Montlaur et al. (2008), the jump operator $[\![\cdot]\!]$ is introduced. Along each portion of the interface Γ, it sums the values of the elements on the left and right, denoted Ω_l and Ω_r, such that

$$[\![\odot]\!] = \odot_l + \odot_r. \tag{2.9}$$

It is important to note that this definition of the jump operator always involves the outward unit normal to a surface, denoted as $[\![\odot \boldsymbol{n}]\!]$. At the interface between elements Ω_l and Ω_r, this implies $[\![\odot \boldsymbol{n}]\!] = \odot_l \boldsymbol{n}_l + \odot_r \boldsymbol{n}_r$, where $\boldsymbol{n}_l$ and $\boldsymbol{n}_r$ are the outward unit normals to $\partial\Omega_l$ and $\partial\Omega_r$, respectively. Furthermore, it should be noted that $\boldsymbol{n}_l = -\boldsymbol{n}_r$ along their interface.

As other DG methods, the HDG method utilises a mixed formulation of the incompressible Navier-Stokes. The system of partial differential equations (2.8) is written as a system of first-order partial differential equations by introducing a new variable $\boldsymbol{L}$, namely

$$
\begin{cases}
\boldsymbol{L} + \sqrt{2\nu}\boldsymbol{\nabla}^{\mathrm{s}}\boldsymbol{u} = \boldsymbol{0} & \text{in } \Omega_e \times (0,T] \quad (2.10a)\\
\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\sqrt{2\nu}\boldsymbol{L} + p\boldsymbol{I}_{\mathrm{n_{sd}}}) + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) = \boldsymbol{s} & \text{in } \Omega_e \times (0,T] \quad (2.10b)\\
\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \times (0,T] \quad (2.10c)\\
\boldsymbol{u} = \boldsymbol{u}_D & \text{on } (\partial\Omega_e \cap \Gamma_D) \times (0,T] \quad (2.10d)\\
((\sqrt{2\nu}\boldsymbol{L} - p\boldsymbol{I}_{\mathrm{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n} = -\boldsymbol{t} & \text{on } (\partial\Omega_e \cap \Gamma_N) \times (0,T] \quad (2.10e)\\
\boldsymbol{u} = \boldsymbol{u}_0 & \text{in } \Omega_e \times \{0\} \quad (2.10f)\\
[\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \boldsymbol{0} & \text{on } \Gamma \times (0,T] \quad (2.10g)\\
[\![((\sqrt{2\nu}\boldsymbol{L} - p\mathbf{I}_{\mathrm{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n}]\!] = \boldsymbol{0} & \text{on } \Gamma \times (0,T]. \quad (2.10h)
\end{cases}
$$

The mixed variable $\boldsymbol{L} := -\sqrt{2\nu}\nabla^{\mathrm{s}}\boldsymbol{u}$ is introduced as an additional unknown, defined as the scaled strain-rate or second-order velocity deformation tensor. By introducing $\mathbf{L}$ as an independent variable, the second-order partial differential equation is decomposed into a system of first-order equations. This decoupling facilitates more flexible numerical discretisation strategies and enables the method to achieve enhanced numerical stability.

The HDG method solves the problem (2.10) in two stages. First, a local problem is considered in each element to express the velocity, pressure and mixed variable in

terms of a new independent variable, $\hat{\boldsymbol{u}}$, which is the trace of the velocity in the mesh skeleton. The local problems can be written as

$$\begin{cases} \boldsymbol{L} + \sqrt{2\nu}\boldsymbol{\nabla}^{\mathrm{s}}\boldsymbol{u} = \boldsymbol{0} & \text{in } \Omega_e \times (0,T] \quad (2.11a) \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\sqrt{2\nu}\boldsymbol{L} + p\boldsymbol{I}_{\mathtt{n_{sd}}}) + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) = \boldsymbol{s} & \text{in } \Omega_e \times (0,T] \quad (2.11b) \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \times (0,T] \quad (2.11c) \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } (\partial\Omega_e \cap \Gamma_D) \times (0,T] \quad (2.11d) \\ \boldsymbol{u} = \hat{\boldsymbol{u}} & \text{on } (\partial\Omega_e \backslash \Gamma_D) \times (0,T] \quad (2.11e) \\ \boldsymbol{u} = \boldsymbol{u}_0 & \text{in } \Omega_e \times \{0\}, \quad (2.11f) \end{cases}$$

and features a pure Dirichlet problem, which means that an extra condition on the pressure needs to be introduced to remove its indeterminacy, for instance

$$\langle p_e, 1\rangle_{\partial\Omega_e} = \rho_e, \qquad e = 1, \ldots, \mathtt{n_{el}} \tag{2.12}$$

where ρ_e represents the scaled mean pressure on the boundary of element Ω_e.

The second stage of the HDG formulation involves a global problem, encompassing the transmission conditions and the Neumann boundary conditions. The local problem is

$$\begin{cases} [\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \boldsymbol{0} & \text{on } \Gamma \times (0,T] \quad (2.13a) \\ [\![((\sqrt{2\nu}\boldsymbol{L} - p\mathbf{I}_{\mathtt{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n}]\!] = \boldsymbol{0} & \text{on } \Gamma \times (0,T] \quad (2.13b) \\ ((\sqrt{2\nu}\boldsymbol{L} - p\boldsymbol{I}_{\mathtt{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n} = -\boldsymbol{t} & \text{on } \Gamma_N \times (0,T]. \quad (2.13c) \end{cases}$$

Given the Dirichlet boundary condition imposed in the local problems (2.11) and the unique definition of the hybrid velocity on each interior face, the continuity of the velocity, imposed by (2.13a), is automatically satisfied. Therefore the global problem

is simply

$$\begin{cases} [\![((\sqrt{2\nu}\boldsymbol{L} - p\mathbf{I}_{\mathtt{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n}]\!] = \mathbf{0} & \text{on } \Gamma\times(0,T] \qquad (2.14a) \\ ((\sqrt{2\nu}\boldsymbol{L} - p\boldsymbol{I}_{\mathtt{n_{sd}}}) - (\boldsymbol{u}\otimes\boldsymbol{u}))\cdot\boldsymbol{n} = -\boldsymbol{t} & \text{on } \Gamma_N\times(0,T]. \qquad (2.14b) \end{cases}$$

Finally, the global problem is completed with the compatibility condition, induced by the free-divergence condition, which is written now in terms of the hybrid velocity, namely

$$\langle\hat{\boldsymbol{u}}\cdot\boldsymbol{n},1\rangle_{\partial\Omega_e\setminus\Gamma_D} + \langle\boldsymbol{u}_D\cdot\boldsymbol{n},1\rangle_{\partial\Omega_e\cap\Gamma_D} = 0. \tag{2.15}$$

2.4 HDG weak formulation

For each element Ω_e, where $e = 1,\ldots,\mathtt{n_{el}}$, the weak formulation of the local problem can be expressed as follows: given $\boldsymbol{u}_D$ on Γ_D and $\hat{\boldsymbol{u}}$ on $\Gamma\cup\Gamma_N$, find $(\boldsymbol{L}_e,\boldsymbol{u}_e,p_e)\in[\mathcal{H}(\mathrm{div};\Omega_e);\mathbb{S}]\times[\mathcal{H}^1(\Omega_e)]^{\mathtt{n_{sd}}}\times\mathcal{H}^1(\Omega_e)$ that satisfies

$$\begin{cases} -(\boldsymbol{G},\boldsymbol{L})_{\Omega_e} + \left(\boldsymbol{\nabla}\cdot(\sqrt{2\nu}\boldsymbol{G}),\boldsymbol{u}\right)_{\Omega_e} \\ \qquad = \left\langle\boldsymbol{G}\boldsymbol{n},\sqrt{2\nu}\boldsymbol{u}_D\right\rangle_{\partial\Omega_e\cap\Gamma_D} + \left\langle\boldsymbol{G}\boldsymbol{n},\sqrt{2\nu}\hat{\boldsymbol{u}}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} & (2.16a) \\ \left(\boldsymbol{w},\dfrac{\partial\boldsymbol{u}}{\partial t}\right)_{\Omega_e} + \left(\boldsymbol{w},\boldsymbol{\nabla}\cdot\left(\sqrt{2\nu}\boldsymbol{L}\right)\right)_{\Omega_e} + (\boldsymbol{w},\boldsymbol{\nabla}p)_{\Omega_e} \\ \qquad + \left\langle\boldsymbol{w},\left(\widehat{\sqrt{2\nu}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}}\right)\boldsymbol{n} - \left(\sqrt{2\nu}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n}\right\rangle_{\partial\Omega_e} & (2.16b) \\ \qquad - (\boldsymbol{\nabla}\boldsymbol{w},\boldsymbol{u}\otimes\boldsymbol{u})_{\Omega_e} + \left\langle\boldsymbol{w},\left(\widehat{\boldsymbol{u}\otimes\boldsymbol{u}}\right)\boldsymbol{n}\right\rangle_{\partial\Omega_e} = (\boldsymbol{w},\boldsymbol{s})_{\Omega_e} \\ (\boldsymbol{\nabla}q,\boldsymbol{u})_{\Omega_e} = \langle q,\boldsymbol{u}_D\cdot\boldsymbol{n}\rangle_{\partial\Omega_e\cap\Gamma_D} + \langle q,\hat{\boldsymbol{u}}\cdot\boldsymbol{n}\rangle_{\partial\Omega_e\setminus\Gamma_D} & (2.16c) \\ \langle p,1\rangle_{\partial\Omega_e} = \rho_e, & (2.16d) \end{cases}$$

for all $(\boldsymbol{G},\boldsymbol{w},q)\in[\mathcal{H}(\mathrm{div};\Omega_e);\mathbb{S}]\times[\mathcal{H}^1(\Omega_e)]^{\mathtt{n_{sd}}}\times\mathcal{H}^1(\Omega_e)$. Here, $[\mathcal{H}(\mathrm{div};\Omega_e);\mathbb{S}]$ denotes the space of square-integrable symmetric tensors $\mathbb{S}$ of order $\mathtt{n_{sd}}$ on Ω_e with square-integrable row-wise divergence.

It is worth noting that integration by parts is done twice in the the momentum equation,

leading to the difference between physical and numerical fluxes in (2.16b).

The numerical trace of the diffusive and convective fluxes is defined in (Cockburn et al., 2009a) and (Giacomini et al., 2020) as

$$\widehat{(\sqrt{2\nu}\boldsymbol{L}+p\mathbf{I}_{\mathtt{nsd}})\boldsymbol{n}} := \begin{cases} (\sqrt{2\nu}\boldsymbol{L}+p\mathbf{I}_{\mathtt{nsd}})\boldsymbol{n}+\tau^d(\boldsymbol{u}-\boldsymbol{u}_D) & \text{on } \partial\Omega_e\cap\Gamma_D \\ (\sqrt{2\nu}\boldsymbol{L}+p\mathbf{I}_{\mathtt{nsd}})\boldsymbol{n}+\tau^d(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{elsewhere} \end{cases} \tag{2.17}$$

and

$$\widehat{(\boldsymbol{u}\otimes\boldsymbol{u})\boldsymbol{n}} := \begin{cases} (\boldsymbol{u}_D\otimes\boldsymbol{u}_D)\boldsymbol{n}+\tau^a(\boldsymbol{u}-\boldsymbol{u}_D) & \text{on } \partial\Omega_e\cap\Gamma_D \\ (\hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}})\boldsymbol{n}_e+\tau^a(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{elsewhere,} \end{cases} \tag{2.18}$$

respectively.

The diffusive and convective stabilisation parameters, τ^d and τ^a, respectively, play a crucial role in the stability, accuracy, and convergence properties of the HDG method.

The diffusive stabilisation parameter τ^d is typically defined as

$$\tau^d = \frac{\kappa\nu}{\ell}, \tag{2.19}$$

where ℓ denotes a characteristic length scale of the problem domain and $\kappa > 0$ is a numerical scaling factor. Extensive numerical experiments have been performed to study the effect of the choice of the diffusive stabilisation; see, for instance, (Giacomini et al., 2020) and references therein.

For the convective term, the stabilisation parameter τ^a is often chosen based on a characteristic velocity of the fluid flow, namely

$$\tau^a = \beta\|\boldsymbol{u}\|_2 \quad \text{or} \quad \tau^a = \beta\|\boldsymbol{u}\|_\infty, \tag{2.20}$$

where $\boldsymbol{u}$ represents the convective velocity field, and $\beta > 0$ is a numerical parameter independent of the Reynolds number. The norms $\|\cdot\|_2$ and $\|\cdot\|_\infty$ can be evaluated locally on individual elements Ω_e or globally on the entire domain Ω, depending on the

specific implementation.

In this work, the diffusive stabilisation parameter is selected as $\kappa = 10$, given the extra accuracy reported in (Giacomini et al., 2020). For the numerical parameter of the convective stabilisation a value of $\beta = 1$ is chosen to ensure that the admissibility condition, introduced in (Cockburn et al., 2009a) to guarantee stability and well-posedness,

$$\min_{\boldsymbol{x}\in\partial\Omega_e} \left\{\tau^d + \tau^a - \hat{\boldsymbol{u}}\cdot\boldsymbol{n}_e\right\} \geq \gamma > 0 \tag{2.21}$$

is satisfied, for some constant γ. Finally, the $\|\cdot\|_2$ norm is used across the computational domain to define the convective stabilisation. In summary, the specific choice of the diffusive stabilisation in this work is

$$\tau^d = \frac{10\nu}{\ell}, \qquad \tau^a = \max_{\boldsymbol{x}\in\Omega} \|u\|_2. \tag{2.22}$$

Introducing the definitions of the numerical traces from (2.17) and (2.18) into the momentum equation yields the weak form of the local problem: for $e = 1, \ldots, \mathtt{n_{el}}$, find $(\boldsymbol{L}_e, \boldsymbol{u}_e, p_e) \in [\mathcal{H}(\mathrm{div};\Omega_e);\mathbb{S}] \times [\mathcal{H}^1(\Omega_e)]^{\mathtt{n_{sd}}} \times \mathcal{H}^1(\Omega_e)$ such that

$$\begin{cases}
-(\boldsymbol{G},\boldsymbol{L})_{\Omega_e} + \left(\boldsymbol{\nabla}\cdot(\sqrt{2\nu}\boldsymbol{G}),\boldsymbol{u}\right)_{\Omega_e} \\
\qquad = \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\boldsymbol{u}_D\right\rangle_{\partial\Omega_e\cap\Gamma_D} + \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\hat{\boldsymbol{u}}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} & \text{(2.23a)} \\
\left(\boldsymbol{w}, \dfrac{\partial\boldsymbol{u}}{\partial t}\right)_{\Omega_e} + \left(\boldsymbol{w}, \boldsymbol{\nabla}\cdot\left(\sqrt{2\nu}\boldsymbol{L}\right)\right)_{\Omega_e} + (\boldsymbol{w},\boldsymbol{\nabla}p)_{\Omega_e} \\
\qquad - (\boldsymbol{\nabla}\boldsymbol{w}, \boldsymbol{u}\otimes\boldsymbol{u})_{\Omega_e} + \langle \boldsymbol{w}, \tau\boldsymbol{u}\rangle_{\partial\Omega_e} = (\boldsymbol{w},\boldsymbol{s})_{\Omega_e} & \text{(2.23b)} \\
\qquad + \langle \boldsymbol{w}, (\tau - \hat{\boldsymbol{u}}\cdot\boldsymbol{n})\,\boldsymbol{u}_D\rangle_{\partial\Omega_e\cap\Gamma_D} + \langle \boldsymbol{w}, (\tau - \hat{\boldsymbol{u}}\cdot\boldsymbol{n})\,\hat{\boldsymbol{u}}\rangle_{\partial\Omega_e\setminus\Gamma_D} \\
(\boldsymbol{\nabla}q, \boldsymbol{u})_{\Omega_e} = \langle q, \boldsymbol{u}_D\cdot\boldsymbol{n}\rangle_{\partial\Omega_e\cap\Gamma_D} + \langle q, \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\rangle_{\partial\Omega_e\setminus\Gamma_D} & \text{(2.23c)} \\
\langle p, 1\rangle_{\partial\Omega_e} = \rho_e, & \text{(2.23d)}
\end{cases}$$

for all $(\boldsymbol{G}, \boldsymbol{w}, q) \in [\mathcal{H}(\mathrm{div};\Omega_e);\mathbb{S}] \times [\mathcal{H}^1(\Omega_e)]^{\mathtt{n_{sd}}} \times \mathcal{H}^1(\Omega_e)$, where the stabilisation parameter is defined as $\tau = \tau^d + \tau^a$. The local problem provides $(\boldsymbol{L}_e, \boldsymbol{u}_e, p_e)$ for $e = 1, \ldots, \mathtt{n_{el}}$, in terms of the global unknowns $\hat{\boldsymbol{u}}$ and $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_{\mathtt{n_{el}}})^T$.

Analogously, substituting the numerical traces defined in (2.17) and (2.18) into the transmission equations, the variational form of the global problem is obtained. Specifically, find $\hat{\boldsymbol{u}} \in \left[\mathcal{H}^{\frac{1}{2}}\left(\Gamma \cup \Gamma_N\right)\right]^{\mathtt{n_{sd}}}$ and $\boldsymbol{\rho} \in \mathbb{R}^{\mathtt{n_{el}}}$ such that holds:

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Big\{ \left\langle \widehat{\boldsymbol{w}}, \left(\sqrt{2\nu}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} + \langle \widehat{\boldsymbol{w}}, \tau \boldsymbol{u} \rangle_{\partial\Omega_e \setminus \Gamma_D} \\ \qquad\qquad - \langle \widehat{\boldsymbol{w}}, \tau \hat{\boldsymbol{u}} \rangle_{\partial\Omega_e \cap \Gamma} - \langle \widehat{\boldsymbol{w}}, (\tau - \hat{\boldsymbol{u}} \cdot \boldsymbol{n})\, \hat{\boldsymbol{u}} \rangle_{\partial\Omega_e \cap \Gamma_N} \Big\} \qquad (2.24a) \\ \qquad\qquad = - \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \langle \widehat{\boldsymbol{w}}, \boldsymbol{t} \rangle_{\partial\Omega_e \cap \Gamma_N} \\ \langle \hat{\boldsymbol{u}} \cdot \boldsymbol{n}, 1 \rangle_{\partial\Omega_e \setminus \Gamma_D} = - \langle \boldsymbol{u}_D \cdot \boldsymbol{n}, 1 \rangle_{\partial\Omega_e \cap \Gamma_D} \quad \text{for } e = 1, \ldots, \mathtt{n_{el}}, \qquad (2.24b) \end{cases}$$

for all $\widehat{\boldsymbol{w}} \in \left[\mathcal{L}_2\left(\Gamma \cup \Gamma_N\right)\right]^{\mathtt{n_{sd}}}$.

2.5 Temporal discretisation

This section presents the temporal discretisation using backward differentiation formulae (BDF) (Curtiss and Hirschfelder, 1952) and explicit first stage, singly diagonally implicit Runge-Kutta (ESDIRK) methods (Kennedy and Carpenter, 2003).

To simplify the presentation, it is assumed that the temporal domain $[0, T]$ is partitioned into equally spaced intervals, and the time step is denoted by Δt. To denote the evaluation of a certain field, for instance, the velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$ at an instant t^n, the notation $\boldsymbol{u}^n$ is used throughout the whole document.

To present the general form of the methods considered in this work, a general scalar ordinary differential equation is considered, namely

$$\frac{dv}{dt} = f(t, v). \qquad (2.25)$$

2.5.1 BDF discretisation of the Navier-Stokes equations

The general form of an s-step BDF method can be expressed as

$$\sum_{j=0}^{s} \alpha_j v^{n+1-j} = \Delta t \beta f(t^{n+1}, v^{n+1}) \tag{2.26}$$

where α_j and β are coefficients specific to each BDF method. In the current implementation first, second and third order BDF methods have been considered. The coefficients for the first-order method (BDF1) are

$$\alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta = 1. \tag{2.27}$$

For the second-order method (BDF2) the coefficients are

$$\alpha_0 = \frac{3}{2}, \quad \alpha_1 = -2, \quad \alpha_2 = \frac{1}{2}, \quad \beta = 1. \tag{2.28}$$

Finally, for the third-order method (BDF3) the coefficients are

$$\alpha_0 = \frac{11}{6}, \quad \alpha_1 = -3, \quad \alpha_2 = \frac{3}{2}, \quad \alpha_3 = -\frac{1}{3}. \tag{2.29}$$

When solving steady problems, the BDF1 is often considered as a relaxation procedure, to facilitate the convergence to the steady state. However, when transient problems are of interest only higher order methods are considered to guarantee time accuracy. Higher-order BDF methods (BDF2 and BDF3) generally offer improved accuracy compared to BDF1, but require additional storage of previous time steps. BDF3, while more accurate, may exhibit slightly reduced stability compared to BDF2 for certain problems (Hundsdorfer and Verwer, 2013).

Applying this general BDF to the weak form of the local problems (2.23) leads to the

semi-discrete local problems

$$
\begin{cases}
-\left(\boldsymbol{G}, \boldsymbol{L}^{n+1}\right)_{\Omega_e} + \left(\boldsymbol{\nabla}\cdot(\sqrt{2\nu}\boldsymbol{G}), \boldsymbol{u}^{n+1}\right)_{\Omega_e} & \\
\qquad = \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\boldsymbol{u}_D^{n+1}\right\rangle_{\partial\Omega_e\cap\Gamma_D} + \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\hat{\boldsymbol{u}}^{n+1}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} & \text{(2.30a)} \\
\left(\boldsymbol{w}, \sum_{j=0}^{s}\alpha_j\boldsymbol{u}^{n+1-j}\right)_{\Omega_e} + \Delta t\beta\Big[\left(\boldsymbol{w}, \boldsymbol{\nabla}\cdot\left(\sqrt{2\nu}\boldsymbol{L}^{n+1}\right)\right)_{\Omega_e} & \\
\qquad + \left(\boldsymbol{w}, \boldsymbol{\nabla}p^{n+1}\right)_{\Omega_e} - \left(\boldsymbol{\nabla}\boldsymbol{w}, \boldsymbol{u}^{n+1}\otimes\boldsymbol{u}^{n+1}\right)_{\Omega_e} + \left\langle \boldsymbol{w}, \tau^n\boldsymbol{u}^{n+1}\right\rangle_{\partial\Omega_e}\Big] & \text{(2.30b)} \\
\qquad = \Delta t\beta\Big[(\boldsymbol{w}, \boldsymbol{s}^{n+1})_{\Omega_e} + \left\langle \boldsymbol{w}, \left(\tau^n - \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{n}\right)\boldsymbol{u}_D^{n+1}\right\rangle_{\partial\Omega_e\cap\Gamma_D} & \\
\qquad + \left\langle \boldsymbol{w}, \left(\tau^n - \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{n}\right)\hat{\boldsymbol{u}}^{n+1}\right\rangle_{\partial\Omega_e\setminus\Gamma_D}\Big] & \\
\left(\boldsymbol{\nabla}q, \boldsymbol{u}^{n+1}\right)_{\Omega_e} = \left\langle q, \boldsymbol{u}_D^{n+1}\cdot\boldsymbol{n}\right\rangle_{\partial\Omega_e\cap\Gamma_D} + \left\langle q, \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{n}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} & \text{(2.30c)} \\
\left\langle p^{n+1}, 1\right\rangle_{\partial\Omega_e} = \rho_e^{n+1}. & \text{(2.30d)}
\end{cases}
$$

It should be noted that in this work the stabilisation parameter is chosen to be evaluated at t^n to simplify the linearisation process.

Similarly, the semi-discrete form of the global problem is

$$
\begin{cases}
\sum_{e=1}^{\mathtt{n_{el}}}\Big\{\left\langle \widehat{\boldsymbol{w}}, \left(\sqrt{2\nu}\boldsymbol{L}^{n+1} + p^{n+1}\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} + \left\langle \widehat{\boldsymbol{w}}, \tau^n\boldsymbol{u}^{n+1}\right\rangle_{\partial\Omega_e\setminus\Gamma_D} & \\
\qquad -\langle \widehat{\boldsymbol{w}}, \tau^n\hat{\boldsymbol{u}}^{n+1}\rangle_{\partial\Omega_e\cap\Gamma} - \left\langle \widehat{\boldsymbol{w}}, \left(\tau^n - \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{n}\right)\hat{\boldsymbol{u}}^{n+1}\right\rangle_{\partial\Omega_e\cap\Gamma_N}\Big\} & \text{(2.31a)} \\
\qquad = -\sum_{e=1}^{\mathtt{n_{el}}}\langle \widehat{\boldsymbol{w}}, \boldsymbol{t}^{n+1}\rangle_{\partial\Omega_e\cap\Gamma_N} & \\
\left\langle \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{n}, 1\right\rangle_{\partial\Omega_e\setminus\Gamma_D} = -\left\langle \boldsymbol{u}_D^{n+1}\cdot\boldsymbol{n}, 1\right\rangle_{\partial\Omega_e\cap\Gamma_D} \quad \text{for } e = 1, \ldots, \mathtt{n_{el}}. & \text{(2.31b)}
\end{cases}
$$

2.5.2 ESDIRK discretisation of the Navier Stokes equations

The general form of an s-stage ESDIRK method is defined as

$$
v^{n+1} = v^n + \Delta t\sum_{i=1}^{s} b_i z_i, \tag{2.32}
$$

where

$$z_i = f(t^n + c_i\Delta t, v^n + \Delta t \sum_{j=1}^{i} a_{ij} z_j), \quad i = 1, \ldots, s, \tag{2.33}$$

and the coefficients c_i, a_{ij}, and b_i define the specific ESDIRK method. The coefficients are typically represented in a Butcher tableau

$$\begin{array}{c|ccccc} c_1 & a_{11} & & & & \\ c_2 & a_{21} & a_{22} & & & \\ c_3 & a_{31} & a_{32} & a_{33} & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & b_3 & \cdots & b_s \end{array} \tag{2.34}$$

For an ESDIRK method, we have $a_{11} = 0$, $a_{ii} = \gamma$ for $i = 2, \ldots, s$, where γ is a constant, $c_1 = 0$ and $a_{ij} = 0$ for $j > i$.

This generalised formulation facilitates the implementation of ESDIRK methods of different order. The specific selection of coefficients c_i, a_{ij}, and b_i determines the order of the accuracy and stability properties of the method. In this work, five distinct ESDIRK methods have been implemented: ESDIRK23, a second-order, three-stage method (Jørgensen et al., 2018); ESDIRK34 Kværnø, a third-order, four-stage method developed by Kværnø (Kværnø, 2004); ESDIRK46, a fourth-order, six-stage method (Kennedy and Carpenter, 2016); ESDIRK68, a sixth-order, eight-stage method (Alamri and Ketcheson, 2024); and ESDIRK816, an eighth-order, sixteen-stage method (Alamri and Ketcheson, 2024). The selection of these methods enables a comprehensive analysis across various order and stage combinations, facilitating the evaluation of the trade-offs between computational cost and solution accuracy.

It is worth mentioning that the implementation of very high-order schemes (6th order and above) in three-dimensional problems warrants practical consideration, as the number of degrees of freedom per element grows rapidly with polynomial order. The

dimension of the HDG global system, while smaller than traditional DG methods, still increases substantially with order, resulting in considerable memory requirements for very high orders. The computational demands of large 3D problems may preclude the use of correspondingly high-order time integration schemes, suggesting that such high spatial orders may be excessive in practice.

Applying this general ESDIRK method to the weak form of the local problems (2.23) leads to the semi-discrete local problems

$$\begin{cases} -(\boldsymbol{G}, \boldsymbol{L}_i)_{\Omega_e} + \left(\boldsymbol{\nabla} \cdot (\sqrt{2\nu}\boldsymbol{G}), \boldsymbol{u}_i\right)_{\Omega_e} \\ \qquad = \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\boldsymbol{u}_{D,i} \right\rangle_{\partial\Omega_e \cap \Gamma_D} + \left\langle \boldsymbol{G}\boldsymbol{n}, \sqrt{2\nu}\hat{\boldsymbol{u}}_i \right\rangle_{\partial\Omega_e \setminus \Gamma_D} & \text{(2.35a)} \\ (\boldsymbol{w}, \boldsymbol{z}_i)_{\Omega_e} + \left(\boldsymbol{w}, \boldsymbol{\nabla} \cdot \left(\sqrt{2\nu}\boldsymbol{L}_i\right)\right)_{\Omega_e} + (\boldsymbol{w}, \boldsymbol{\nabla} p_i)_{\Omega_e} \\ \qquad - (\boldsymbol{\nabla}\boldsymbol{w}, \boldsymbol{u}_i \otimes \boldsymbol{u}_i)_{\Omega_e} + \langle \boldsymbol{w}, \tau^n \boldsymbol{u}_i \rangle_{\partial\Omega_e} \\ \qquad = (\boldsymbol{w}, \boldsymbol{s}_i)_{\Omega_e} + \langle \boldsymbol{w}, (\tau^n - \hat{\boldsymbol{u}}_i \cdot \boldsymbol{n})\, \boldsymbol{u}_{D,i} \rangle_{\partial\Omega_e \cap \Gamma_D} \\ \qquad + \langle \boldsymbol{w}, (\tau^n - \hat{\boldsymbol{u}}_i \cdot \boldsymbol{n})\, \hat{\boldsymbol{u}}_i \rangle_{\partial\Omega_e \setminus \Gamma_D} & \text{(2.35b)} \\ (\boldsymbol{\nabla} q, \boldsymbol{u}_i)_{\Omega_e} = \langle q, \boldsymbol{u}_{D,i} \cdot \boldsymbol{n} \rangle_{\partial\Omega_e \cap \Gamma_D} + \langle q, \hat{\boldsymbol{u}}_i \cdot \boldsymbol{n} \rangle_{\partial\Omega_e \setminus \Gamma_D} & \text{(2.35c)} \\ \langle p_i, 1 \rangle_{\partial\Omega_e} = \rho_{e,i}. & \text{(2.35d)} \end{cases}$$

and the semi-discrete global problem

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Big\{ \left\langle \widehat{\boldsymbol{w}}, \left(\sqrt{2\nu}\boldsymbol{L}_i + p_i \mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} + \langle \widehat{\boldsymbol{w}}, \tau^n \boldsymbol{u}_i \rangle_{\partial\Omega_e \setminus \Gamma_D} \\ \qquad - \langle \widehat{\boldsymbol{w}}, \tau^n \hat{\boldsymbol{u}}_i \rangle_{\partial\Omega_e \cap \Gamma} - \langle \widehat{\boldsymbol{w}}, (\tau^n - \hat{\boldsymbol{u}}_i \cdot \boldsymbol{n})\, \hat{\boldsymbol{u}}_i \rangle_{\partial\Omega_e \cap \Gamma_N} \Big\} & \text{(2.36a)} \\ \qquad = - \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \langle \widehat{\boldsymbol{w}}, \boldsymbol{t}_i \rangle_{\partial\Omega_e \cap \Gamma_N} \\ \langle \hat{\boldsymbol{u}}_i \cdot \boldsymbol{n}, 1 \rangle_{\partial\Omega_e \setminus \Gamma_D} = - \langle \boldsymbol{u}_{D,i} \cdot \boldsymbol{n}, 1 \rangle_{\partial\Omega_e \cap \Gamma_D} \quad \text{for } e = 1, \ldots, \mathtt{n_{el}}. & \text{(2.36b)} \end{cases}$$

to be solved at each stage, for $i = 1, \ldots, s$.

As mentioned earlier, in this work the stabilisation parameter is chosen to be evaluated at t^n to simplify the linearisation process.

The solution at time t^{n+1} is then obtained using the values of each stage as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{z}_i, \tag{2.37}$$

where $\boldsymbol{z}_i$ are computed as in equation (2.33).

The gradient of velocity $\boldsymbol{L}$, pressure p, and the hybrid variable $\hat{\boldsymbol{u}}$ are computed at each stage of the ESDIRK method by solving the coupled system of equations derived from the semi-discrete local and global problems (2.35) and (2.36). The final values at t^{n+1} are then obtained from the last stage

$$\boldsymbol{L}^{n+1} = \boldsymbol{L}_s, \qquad p^{n+1} = p_s, \qquad \hat{\boldsymbol{u}}^{n+1} = \hat{\boldsymbol{u}}_s. \tag{2.38}$$

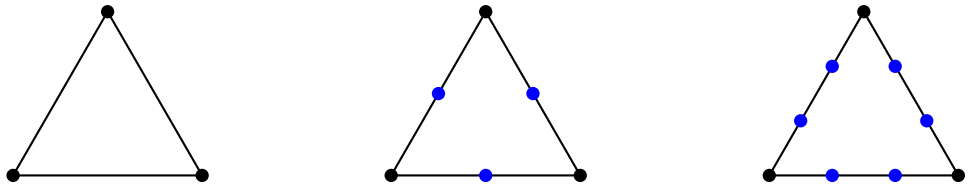
2.6 Spatial discretisation

High-order methods utilise elements with additional nodes to achieve increased accuracy in both solution and geometric approximation. The spatial discretisation is performed using isoparametric elements. A mapping is established between a reference element and a face, $\widetilde{\Omega}$ and $\widetilde{\Gamma}$, respectively, and the physical elements and faces. The elemental isoparametric mapping can be written as

$$\begin{aligned} \boldsymbol{\varphi} \ : \widetilde{\Omega} \subset \mathbb{R}^{\mathtt{n_{sd}}} &\longrightarrow \Omega_e \subset \mathbb{R}^{\mathtt{n_{sd}}} \\ \boldsymbol{\xi} &\longmapsto \boldsymbol{\varphi}(\boldsymbol{\xi}) := \sum_{j=1}^{\mathtt{n_{en}}} \boldsymbol{x}_j N_j(\boldsymbol{\xi}), \end{aligned} \tag{2.39}$$

where $\mathtt{n_{en}}$ is the number of element nodes, $\{\boldsymbol{x}_j\}_{j=1,\ldots,\mathtt{n_{en}}}$ are the nodal coordinates of Ω_e and $\{N_j\}_{j=1,\ldots,\mathtt{n_{en}}}$ are the polynomial shape functions of order k defined in the reference element.

For elements adjacent to curved boundaries, standard straight-sided elements are insufficient. The isoparametric formulation allows curved elements to accurately represent nonlinear geometries through appropriate node placement, as shown in Figure 2.2.



(a) (b) (c)

Figure 2.1: High-order triangular elements: (a) Linear element with vertex nodes, (b) Quadratic element with additional edge nodes (blue), (c) Cubic element with edge nodes. Additional nodes enable higher-order polynomial approximations.

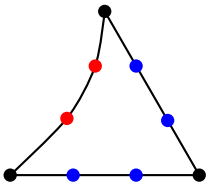


Figure 2.2: Cubic triangular element with curved boundary. Edge nodes (blue) and additional nodes on the curved edge (red) enable accurate geometric representation.

For non-linear mappings where $k > 1$, the inverse mapping requires solving:

$$\sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \boldsymbol{x}_j N_j(\boldsymbol{\xi}) - \boldsymbol{x} = \boldsymbol{0} \tag{2.40}$$

Using a Taylor series expansion and truncating after the first derivative yields:

$$\boldsymbol{R}_{l+1} = \boldsymbol{R}_l + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}_l} \Delta\boldsymbol{\xi} \tag{2.41}$$

where $\Delta\boldsymbol{\xi} = \boldsymbol{\xi}_{l+1} - \boldsymbol{\xi}_l$ and $\boldsymbol{R}_l = \sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \boldsymbol{x}_j N_j(\boldsymbol{\xi}_l) - \boldsymbol{x}$ is the residual at iteration l.

Setting $\boldsymbol{R}_{l+1} = \boldsymbol{0}$ and rearranging for $\boldsymbol{\xi}_{l+1}$ gives:

$$\boldsymbol{\xi}_{l+1} = \boldsymbol{\xi}_l - \left(\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}_l}\right)^{-1} \boldsymbol{R}_l \tag{2.42}$$

Substituting the expression for $\boldsymbol{R}$ and simplifying the tangent matrix yields the Newton-Raphson iteration:

$$\boldsymbol{\xi}_{l+1} = \boldsymbol{\xi}_l - \left(\boldsymbol{x}_j \otimes \nabla_\xi N_j(\boldsymbol{\xi}_l)\right)^{-1} \boldsymbol{R}_l \tag{2.43}$$

The initial guess is taken as the mean of the nodal coordinates:

$$\boldsymbol{\xi}_0 = \frac{1}{\mathtt{n}_{\mathtt{en}}} \sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \boldsymbol{\xi}_j \tag{2.44}$$

For linear elements ($k = 1$), the mapping converges in one iteration as the residual is linear and the tangent matrix is constant, since the shape functions are linear in $\boldsymbol{\xi}$. For high-order elements ($k > 1$), the shape functions can be non-linear in $\boldsymbol{\xi}$, making the tangent matrix $\boldsymbol{\xi}$-dependent and requiring multiple Newton-Raphson iterations.

Similarly, the isoparametric mapping for a face can be written as

$$\begin{aligned} \boldsymbol{\psi} \ : \widetilde{\Gamma} \subset \mathbb{R}^{\mathtt{n}_{\mathtt{sd}}-1} &\longrightarrow \Gamma_e \subset \mathbb{R}^{\mathtt{n}_{\mathtt{sd}}} \\ \boldsymbol{\eta} &\longmapsto \boldsymbol{\psi}(\boldsymbol{\eta}) := \sum_{j=1}^{\mathtt{n}_{\mathtt{fn}}} \boldsymbol{x}_j \hat{N}_j(\boldsymbol{\eta}), \end{aligned} \tag{2.45}$$

where $\mathtt{n}_{\mathtt{fn}}$ is the number of face nodes, $\{\boldsymbol{x}_j\}_{j=1,\dots,\mathtt{n}_{\mathtt{en}}}$ are the nodal coordinates of Ω_e and $\{\hat{N}_j\}_{j=1,\dots,\mathtt{n}_{\mathtt{en}}}$ are the polynomial shape functions of order k defined on the reference face.

With the link established between reference and physical elements/faces, the polynomial functional approximation is defined on the reference element/face, using the same shape functions introduced in the geometrical mappings, namely

$$\boldsymbol{u}(\boldsymbol{\xi}) \simeq \boldsymbol{u}_h(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \mathbf{u}_j N_j(\boldsymbol{\xi}) \quad \in \left[\{v \in \mathcal{L}^2(\Omega);\ v|_{\Omega_e} \in \mathcal{P}^k(\widetilde{\Omega})\}\right]^{\mathtt{n}_{\mathtt{sd}}} \tag{2.46a}$$

$$p(\boldsymbol{\xi}) \simeq p_h(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \mathrm{p}_j N_j(\boldsymbol{\xi}) \quad \in \{q \in \mathcal{L}^2(\Omega);\ q|_{\Omega_e} \in \mathcal{P}^k(\widetilde{\Omega})\} \tag{2.46b}$$

$$\boldsymbol{L}(\boldsymbol{\xi}) \simeq \boldsymbol{L}_h(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n}_{\mathtt{en}}} \mathbf{L}_j N_j(\boldsymbol{\xi}) \quad \in \left[\{v \in \mathcal{L}^2(\Omega);\ v|_{\Omega_e} \in \mathcal{P}^k(\widetilde{\Omega})\}\right]^{\mathtt{n}_{\mathtt{sd}} \times \mathtt{n}_{\mathtt{sd}}} \tag{2.46c}$$

$$\hat{\boldsymbol{u}}(\boldsymbol{\eta}) \simeq \hat{\boldsymbol{u}}_h(\boldsymbol{\eta}) = \sum_{j=1}^{\mathtt{n}_{\mathtt{fn}}} \hat{\mathbf{u}}_j \hat{N}_j(\boldsymbol{\eta}) \quad \in \left[\{v \in \mathcal{L}^2(\Gamma);\ v|_{\Gamma_e} \in \mathcal{P}^{\hat{k}}(\widetilde{\Gamma})\}\right]^{\mathtt{n}_{\mathtt{sd}}}. \tag{2.46d}$$

where $\mathbf{u}_j$, p_j , $\mathbf{L}_j$ and $\hat{\mathbf{u}}_j$ are the nodal values of the velocity, pressure, mixed variable and hybrid velocity, respectively.

Introducing the approximations of Equation (2.46) into the semi-discrete form of the local problems of Equation (2.30) and selecting the space of weighting functions equal to the space spanned by the shape functions, leads to the definition of the discrete elemental local residuals

$$\mathbf{R}^e_{L,i} := \left(N_i, \boldsymbol{L}_h^{n+1}\right)_{\Omega_e} - \left(\sqrt{2\nu}, \boldsymbol{\nabla} N_i \otimes \boldsymbol{u}_h^{n+1}\right)_{\Omega_e} + \left\langle \sqrt{2\nu} N_i, \boldsymbol{u}_D^{n+1} \otimes \boldsymbol{n} \right\rangle_{\partial\Omega_e \cap \Gamma_D} + \left\langle \sqrt{2\nu} N_i, \hat{\boldsymbol{u}}_h^{n+1} \otimes \boldsymbol{n} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} \tag{2.47a}$$

$$\begin{aligned}\mathbf{R}^e_{u,i} := & \left(N_i, \sum_{j=0}^{s} \alpha_j \boldsymbol{u}_h^{n+1-j} \right)_{\Omega_e} + \Delta t \beta \Bigg[\left(N_i, \boldsymbol{\nabla} \cdot \left(\sqrt{2\nu} \boldsymbol{L}_h^{n+1}\right)\right)_{\Omega_e} + \left(N_i, \boldsymbol{\nabla} p_h^{n+1}\right)_{\Omega_e} \\ & - \left(\boldsymbol{\nabla} N_i, \boldsymbol{u}_h^{n+1} \otimes \boldsymbol{u}_h^{n+1}\right)_{\Omega_e} + \left\langle N_i, \tau^n \boldsymbol{u}_h^{n+1} \right\rangle_{\partial\Omega_e} - \left(N_i, \boldsymbol{s}^{n+1}\right)_{\Omega_e} \\ & - \left\langle N_i, \left(\tau^n - \hat{\boldsymbol{u}}_h^{n+1} \cdot \boldsymbol{n}\right) \boldsymbol{u}_D^{n+1} \right\rangle_{\partial\Omega_e \cap \Gamma_D} - \left\langle N_i, \left(\tau^n - \hat{\boldsymbol{u}}_h^{n+1} \cdot \boldsymbol{n}\right) \hat{\boldsymbol{u}}_h^{n+1} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} \Bigg]\end{aligned} \tag{2.47b}$$

$$\mathbf{R}^e_{p,i} := \left(\boldsymbol{\nabla} N_i, \boldsymbol{u}_h^{n+1}\right)_{\Omega_e} - \left\langle N_i, \boldsymbol{u}_D^{n+1} \cdot \boldsymbol{n} \right\rangle_{\partial\Omega_e \cap \Gamma_D} - \left\langle N_i, \hat{\boldsymbol{u}}_h^{n+1} \cdot \boldsymbol{n} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} \tag{2.47c}$$

$$\mathbf{R}^e_{\zeta,i} := \left\langle p_h^{n+1}, 1 \right\rangle_{\partial\Omega_e} - \rho_e^{n+1}. \tag{2.47d}$$

Analogously, introducing the approximations of Equation (2.46) into the semi-discrete form of the global problem of Equation (2.31) leads to the definition of the discrete global residuals

$$\begin{aligned}\mathbf{R}_{\hat{u},i} := \sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ & \left\langle \widehat{N}_i, \left(\sqrt{2\nu} \boldsymbol{L}_h^{n+1} + p_h^{n+1} \mathbf{I}_{\mathtt{n_{sd}}}\right) \boldsymbol{n} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} + \left\langle \widehat{N}_i, \tau^n \boldsymbol{u}_h^{n+1} \right\rangle_{\partial\Omega_e \setminus \Gamma_D} \\ & - \left\langle \widehat{N}_i, \tau^n \hat{\boldsymbol{u}}_h^{n+1} \right\rangle_{\partial\Omega_e \cap \Gamma} - \left\langle \widehat{N}_i, \left(\tau^n - \hat{\boldsymbol{u}}_h^{n+1} \cdot \boldsymbol{n}\right) \hat{\boldsymbol{u}}_h^{n+1} \right\rangle_{\partial\Omega_e \cap \Gamma_N} \\ & + \left\langle \widehat{N}_i, \boldsymbol{t}^{n+1} \right\rangle_{\partial\Omega_e \cap \Gamma_N} \Bigg\}\end{aligned} \tag{2.48a}$$

$$\mathbf{R}^e_{\rho,i} := \left\langle \hat{\boldsymbol{u}}_h^{n+1} \cdot \boldsymbol{n}, 1 \right\rangle_{\partial\Omega_e \setminus \Gamma_D} + \left\langle \boldsymbol{u}_D^{n+1} \cdot \boldsymbol{n}, 1 \right\rangle_{\partial\Omega_e \cap \Gamma_D}. \tag{2.48b}$$

The above residuals correspond to the temporal discretisation using BDF schemes, but almost identical expressions are obtained for each stage of the ESDIRK method given by the local and global problems of Equations (2.35) and (2.36).

2.7 Newton-Raphson linearisation

The Newton-Raphson method is used to linearise the residuals of local and global problems given by Equations (2.47) and (2.48), respectively. By truncating the Taylor series expansion to its first order, the resulting linear system that needs to be solved in each iteration of the Newton-Raphson iteration, m, is given by

$$\begin{bmatrix} \mathbf{T}_{UU} & \mathbf{T}_{U\Lambda} \\ \mathbf{T}_{\Lambda U} & \mathbf{T}_{\Lambda\Lambda} \end{bmatrix}^{n,m} \begin{Bmatrix} \Delta\mathbf{U} \\ \Delta\boldsymbol{\Lambda} \end{Bmatrix}^{n,m} = - \begin{Bmatrix} \mathbf{R}_U \\ \mathbf{R}_\Lambda \end{Bmatrix}^{n,m} \tag{2.49}$$

where,

$$\begin{aligned}
\mathbf{T}_{UU} &= \begin{bmatrix} \mathbf{T}_{LL} & \mathbf{T}_{Lu} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{Lu}^T & \mathbf{T}_{uu} & \mathbf{T}_{pu}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{pu} & \mathbf{0} & \mathbf{t}_{\rho p}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{t}_{\rho p} & 0 \end{bmatrix}, \quad \mathbf{T}_{U\Lambda} = \begin{bmatrix} \mathbf{T}_{L\hat{u}} & 0 \\ \mathbf{T}_{u\hat{u}} & 0 \\ \mathbf{T}_{p\hat{u}} & 0 \\ 0 & 1 \end{bmatrix}, \\
\mathbf{T}_{\Lambda U} &= \begin{bmatrix} \mathbf{T}_{\hat{u}L} & \mathbf{T}_{\hat{u}u} & \mathbf{T}_{\hat{u}p} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_{\Lambda\Lambda} = \begin{bmatrix} \mathbf{T}_{\hat{u}\hat{u}} & 0 \\ \mathbf{T}_{p\hat{u}} & 0 \end{bmatrix}, \\
\mathbf{U} &= \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \\ \mathbf{p} \\ \zeta \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \end{Bmatrix}, \quad \mathbf{R}_U = \begin{Bmatrix} \mathbf{R}_L \\ \mathbf{R}_u \\ \mathbf{R}_p \\ \mathbf{R}_\zeta \end{Bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_\rho \end{Bmatrix}.
\end{aligned} \tag{2.50}$$

In the above expressions, $\mathbf{T}_{ab}$ denotes the tangent matrix obtained by assembling the contributions $(\mathbf{T}_{ab})_{e,i} := \partial \mathbf{R}_{i,a}^e / \partial b$ and $\Delta \circledcirc^{n,m} = \circledcirc^{n,m+1} - \circledcirc^{n,m}$ is the increment of the solution from the Newton-Raphson iteration m to $m+1$.

Given the element-by-element structure of the block $\mathbf{T}_{UU}$ the hybridisation process is applied to reduce this system to one involving only the degree of freedom of the hybrid

velocity and the mean value of the pressure in each element, $\Delta\mathbf{\Lambda}$,

$$\mathbb{K}^{n,m}\Delta\Lambda^{n,m} = \mathbb{F}^{n,m} \tag{2.51}$$

where

$$\mathbb{K}^{n,m} = \mathbf{T}^{n,m}_{\Lambda\Lambda} - \mathbf{T}^{n,m}_{\Lambda U}(\mathbf{T}^{n,m}_{UU})^{-1}\mathbf{T}^{n,m}_{U\Lambda} \tag{2.52}$$

$$\mathbb{F}^{n,m} = -\mathbf{R}^{n,m}_{\Lambda} + \mathbf{T}^{n,m}_{\Lambda U}(\mathbf{T}^{n,m}_{UU})^{-1}\mathbf{R}^{n,m}_{U}. \tag{2.53}$$

The global linear system of equation (2.51) is solved using high-performance numerical libraries. For systems with fewer than one million degrees of freedom, the direct solver PARDISO from the Intel Math Kernel Library (MKL) is employed (Schenk and Gärtner, 2004). This solver is chosen for its robustness and efficiency in handling sparse, unsymmetric matrices. For larger systems, the multi-core PETSc (Portable, Extensible Toolkit for Scientific Computation) implementation is utilised (Balay et al., 2019), specifically employing the MUMPS (MUltifrontal Massively Parallel sparse direct Solver) package (Amestoy et al., 2001). This combination leverages the advantages of distributed memory parallelism and the efficiency of multifrontal algorithms for large-scale sparse linear systems. The tolerance for convergence of the linear system residual is set to 10^{-12} in all examples presented in this work.

After solving the global problem to obtain $\Delta\Lambda^{n,m}$, the increment $\Delta U^{n,m}$ which involves the velocity, pressure and mixed variable in each element can be computed as

$$\mathbf{T}^{n,m}_{UU}\Delta U^{n,m} = -\mathbf{R}^{n,m}_{U} - \mathbf{T}^{n,m}_{U\Lambda}\Delta\Lambda^{n,m}. \tag{2.54}$$

It is worth emphasising that the solution of the local problems of Equation (2.54) can be done element by element and in parallel, without communication required, as the problems are independent.

The convergence of the Newton-Raphson method is determined by examining both the

relative increments of the solution variables and the relative residuals.

Convergence of the relative increments involves checking that

$$\max\{\mathcal{R}_L, \mathcal{R}_u, \mathcal{R}_p, \mathcal{R}_{\hat{u}}, \mathcal{R}_\rho\} < \varepsilon_{\texttt{NR}} \tag{2.55}$$

where

$$\begin{aligned}
\mathcal{R}_L &:= \frac{\|\mathbf{L}^{n,m+1} - \mathbf{L}^{n,m}\|_\infty}{\max\{\|\mathbf{L}^{n,m+1}\|_\infty, \epsilon_{\text{den}}\}}, \qquad & \mathcal{R}_u &:= \frac{\|\mathbf{u}^{n,m+1} - \mathbf{u}^{n,m}\|_\infty}{\max\{\|\mathbf{u}^{n,m+1}\|_\infty, \epsilon_{\text{den}}\}}, \\
\mathcal{R}_p &:= \frac{\|\mathbf{p}^{n,m+1} - \mathbf{p}^{n,m}\|_\infty}{\max\{\|\mathbf{p}^{n,m+1}\|_\infty, \epsilon_{\text{den}}\}}, \qquad & \mathcal{R}_{\hat{u}} &:= \frac{\|\hat{\mathbf{u}}^{n,m+1} - \hat{\mathbf{u}}^{n,m}\|_\infty}{\max\{\|\hat{\mathbf{u}}^{n,m+1}\|_\infty, \epsilon_{\text{den}}\}}, \\
\mathcal{R}_\rho &:= \frac{\|\boldsymbol{\rho}^{n,m+1} - \boldsymbol{\rho}^{n,m}\|_\infty}{\max\{\|\boldsymbol{\rho}^{n,m+1}\|_\infty, \epsilon_{\text{den}}\}},
\end{aligned} \tag{2.56}$$

and ϵ_{den} is a regularisation parameter introduced to prevent division by zero and, in all examples, it is taken as $\epsilon_{\text{den}} = 10^{-7}$.

Similarly, convergence of the residuals involves checking that

$$\max\left\{\frac{\|\mathbf{R}_L\|_\infty}{\widetilde{R}_L}, \frac{\|\mathbf{R}_u\|_\infty}{\widetilde{R}_u}, \frac{\|\mathbf{R}_p\|_\infty}{\widetilde{R}_p}, \frac{\|\mathbf{R}_{\hat{u}}\|_\infty}{\widetilde{R}_{\hat{u}}}, \frac{\|\mathbf{R}_\rho\|_\infty}{\widetilde{R}_\rho}\right\} < \varepsilon_{\texttt{NR}} \tag{2.57}$$

where the normalisation term $\widetilde{R}_\square := \max\{\|\widetilde{\mathbf{R}}_\square\|_\infty, \epsilon_{\text{den}}\}$ is computed from the known contributions of the corresponding residual, $\widetilde{\mathbf{R}}_\square$, namely Dirichlet, Neumann and source term contributions.

The Newton-Raphson method is considered to have converged when both conditions for the relative increment and the relative residual criteria are satisfied, as given by Equations (2.55) and (2.57). In all numerical examples, the tolerance to stop the Newton-Raphson iterations is taken as $\varepsilon_{\texttt{NR}} = 0.5 \times 10^{-10}$.

2.8 Verification examples

This section presents a series of numerical tests to verify the correct implementation of the Fortran 90 code to solve the incompressible Navier-Stokes equations using the HDG formulation described in this chapter. To this end, steady and transient examples with a known analytical solution are considered. The optimal convergence of the $\mathcal{L}^2(\Omega)$ norm of the error for velocity, pressure, and the mixed variable, when the mesh is refined or the time step decreased.

2.8.1 Kovasznay Flow

The first numerical test considers the Kovasznay flow (Kovasznay, 1948), a well-known analytical solution to the incompressible Navier-Stokes equations. This problem provides an excellent benchmark for assessing the accuracy and convergence properties of numerical methods. The computational domain is a unit square, $\Omega = [0,1]^2$, and the analytical solution is given by

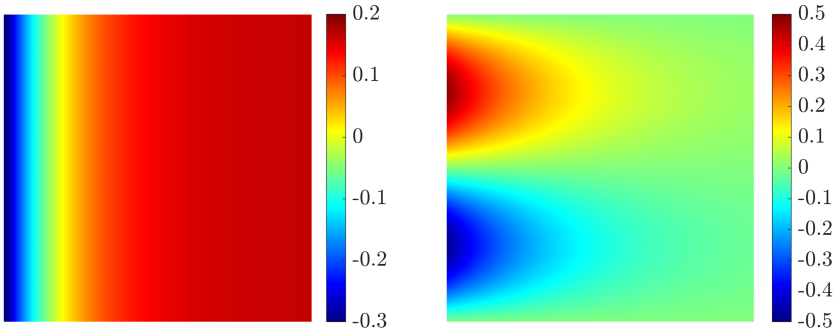
$$\boldsymbol{u}(\boldsymbol{x}) = \begin{Bmatrix} 1 - \exp(2\lambda x)\cos(2\pi y) \\ \frac{\lambda}{2\pi}\exp(2\lambda x)\sin(2\pi y) \end{Bmatrix}, \quad p(\boldsymbol{x}) = -\frac{1}{2}\exp(4\lambda x) + C, \tag{2.58}$$

where $\lambda = \dfrac{Re}{2} - \sqrt{\dfrac{Re^2}{4} + 4\pi^2}$ and $C = \dfrac{1}{8}\left[1 + \exp(4\lambda) - \dfrac{1}{2\lambda}(1 - \exp(4\lambda))\right]$.

The Reynolds number is taken as $Re = 100$ and the analytical solution is shown in Figure 2.3.

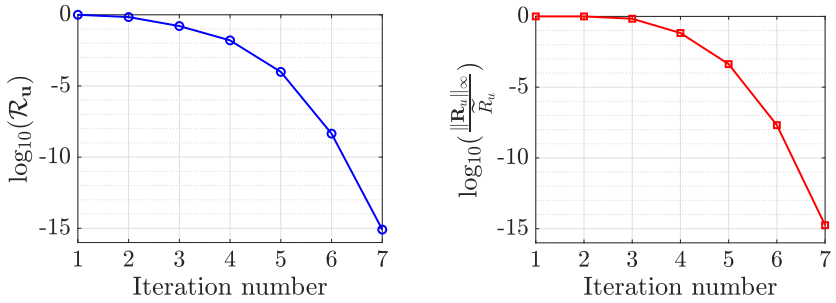
A Neumann boundary condition, corresponding to the exact solution, is imposed on the bottom part of the boundary, $\Gamma_N = \{(x,y) \in \Omega \mid y = 0\}$, while Dirichlet boundary conditions, corresponding to the exact velocity, are imposed on $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

Four uniform meshes are considered, with 16, 64, 256, and 1,024 triangular elements, respectively, as shown in Figure 2.5.



(a) Pressure (b) Velocity magnitude

Figure 2.3: Kovasznay Flow: Pressure and velocity fields for $Re = 100$ in the domain $\Omega = [0, 1]^2$.

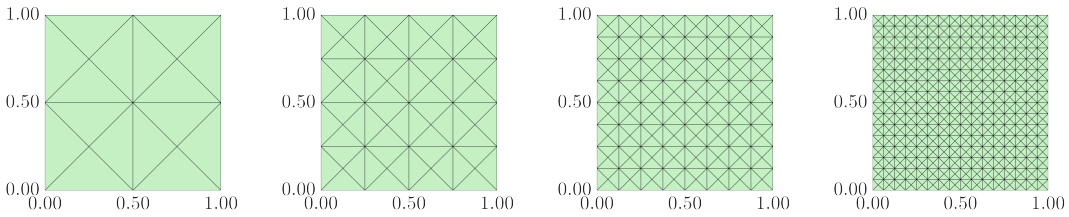


(a) Relative increment (b) Relative residual

Figure 2.4: Kovasznay Flow: Convergence of the NR method, relative increment, and relative residual versus the iteration number on the velocity

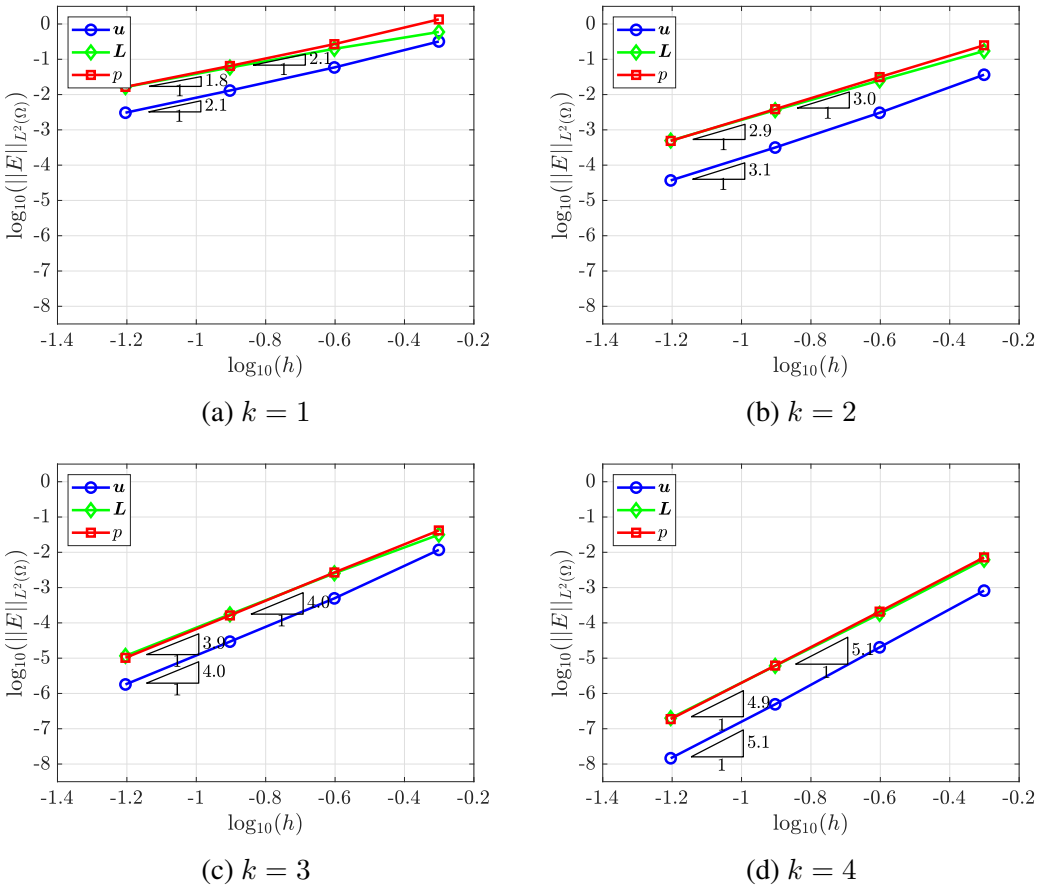
Figure 2.4 illustrates the convergence behaviour of the Newton-Raphson. The plot shows the logarithm of the relative increment and relative residual for the velocity computed from Equations (2.55) and (2.57) versus the iteration number of the Newton-Raphson method for the Mesh 1 of 16 elements. The Figure 2.4 demonstrates the rapid convergence of the Newton-Raphson (NR) method. Both the relative increment (a) and the relative residual (b) decrease exponentially, reaching approximately 10^{-15} after only 7 iterations. This indicates quadratic convergence, characteristic of the NR method. The smooth curves suggest stable convergence throughout the process. The similar shapes of both plots imply a strong correlation between the reduction in solution updates and the minimisation of the residual.

Figure 2.6 shows the $\mathcal{L}^2(\Omega)$ norm of the error of the velocity, pressure and mixed variable as a function of the characteristic element size h for a degree of approximation ranging from $k = 1$ up to $k = 4$. The expected $k + 1$ convergence rate can be observed



(a) Mesh 1 (b) Mesh 2 (c) Mesh 3 (d) Mesh 4

Figure 2.5: Triangular meshes of the domain $\Omega = [0,1]^2$ used to test the optimal convergence properties of the HDG method.



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

Figure 2.6: Kovasznay Flow: $\mathcal{L}_2(\Omega)$ norm of the error for the velocity, pressure and mixed variable as a function of the characteristic element size h, for different degrees of approximation.

for all the variables and degree of approximation. The results show not only the same rate of convergence for all the variables, but also a similar error for all the variables in a fixed mesh. For sufficiently regular solutions, this optimal convergence behaviour is consistently observed across different mesh sizes and polynomial degrees..Comparing the results for low and high order approximations provide evidence of the potential of high order approximations. For example, the solution obtained in the coarsest mesh

using $k = 4$, with only 16 elements, is more accurate than the solution obtained in the finest mesh using $k = 1$, with 1,024 elements. As the dominant cost of the HDG method is driven by the solution of the global problem, it is instructive to consider the size of the global problem as a measure of the cost to compare low and high order approximations. The solution of the problem with $k = 1$ on a mesh of 1,024 elements requires the solution of global problems of 4,128 degrees of freedom, whereas the solution of the problem with $k = 4$ on a mesh of 16 elements requires the solution of global problems of 146 degrees of freedom.

2.8.2 Ethier-Steinmann flow

The second numerical test considers the Ethier-Steinman flow (Ethier and Steinman, 1994) in $\Omega = [0, 1]^3$, which involves a known time-dependent analytical solution for the incompressible Navier-Stokes equations. The solution is given by

$$\boldsymbol{u}(\boldsymbol{x}) = -a \begin{Bmatrix} e^{ax}\sin(ay - dz) + e^{az}\cos(ax - dy) \\ e^{ay}\sin(az - dx) + e^{ax}\cos(ay - dz) \\ e^{az}\sin(ax - dy) + e^{ay}\cos(az - dx) \end{Bmatrix} e^{-d^2 t} \tag{2.59a}$$

$$\begin{aligned} p(\boldsymbol{x}) = -\frac{a^2}{2}\Big[& e^{2ax} + e^{2ay} + e^{2az} + 2\sin(ax - dy)\cos(az - dx)e^{a(y+z)} \\ & + 2\sin(ay - dz)\cos(ax - dy)e^{a(z+x)} \\ & + 2\sin(az - dx)\cos(ay - dz)e^{a(x+y)}\Big] e^{-2d^2 t} \end{aligned} \tag{2.59b}$$

where $a = \frac{\pi}{4}$ and $d = \frac{\pi}{2}$. The final time is $T_{\texttt{end}} = 0.05$ and the time step $\Delta t = 0.01$, which is selected so the temporal error does not affect the spatial convergence. As in the previous example, to test the correct implementation of the Neumann and Dirichlet boundary conditions, a Neumann boundary condition, corresponding to the exact solution, is imposed on the bottom part of the boundary, $\Gamma_N = \{(x, y, z) \in \Omega \mid z = 0\}$, while Dirichlet boundary conditions, corresponding to the exact velocity, are imposed on $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

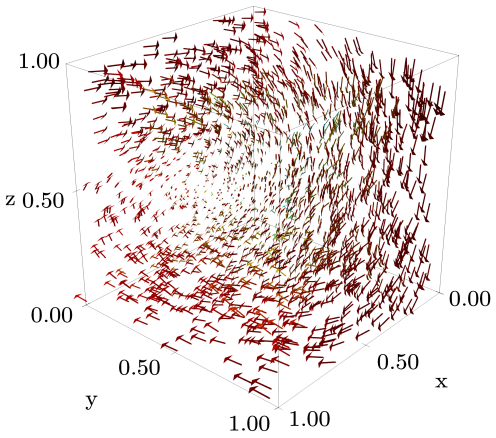
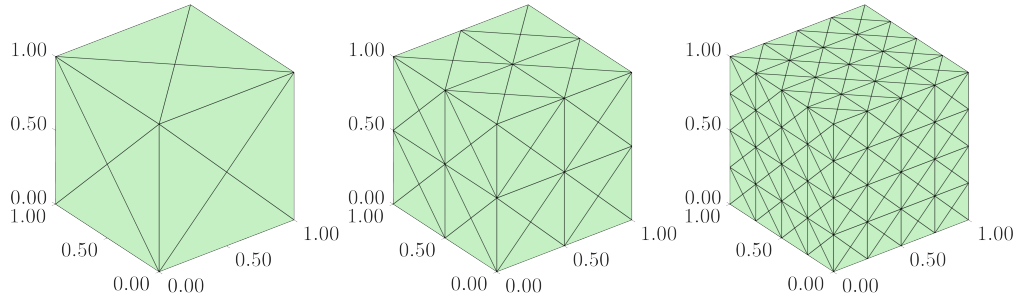


Figure 2.7: Ethier-Steinmann flow: velocity field for $a = \frac{\pi}{4}$ and $d = \frac{\pi}{2}$ at time $t = 0$.



(a) Mesh 1 (b) Mesh 2 (c) Mesh 3

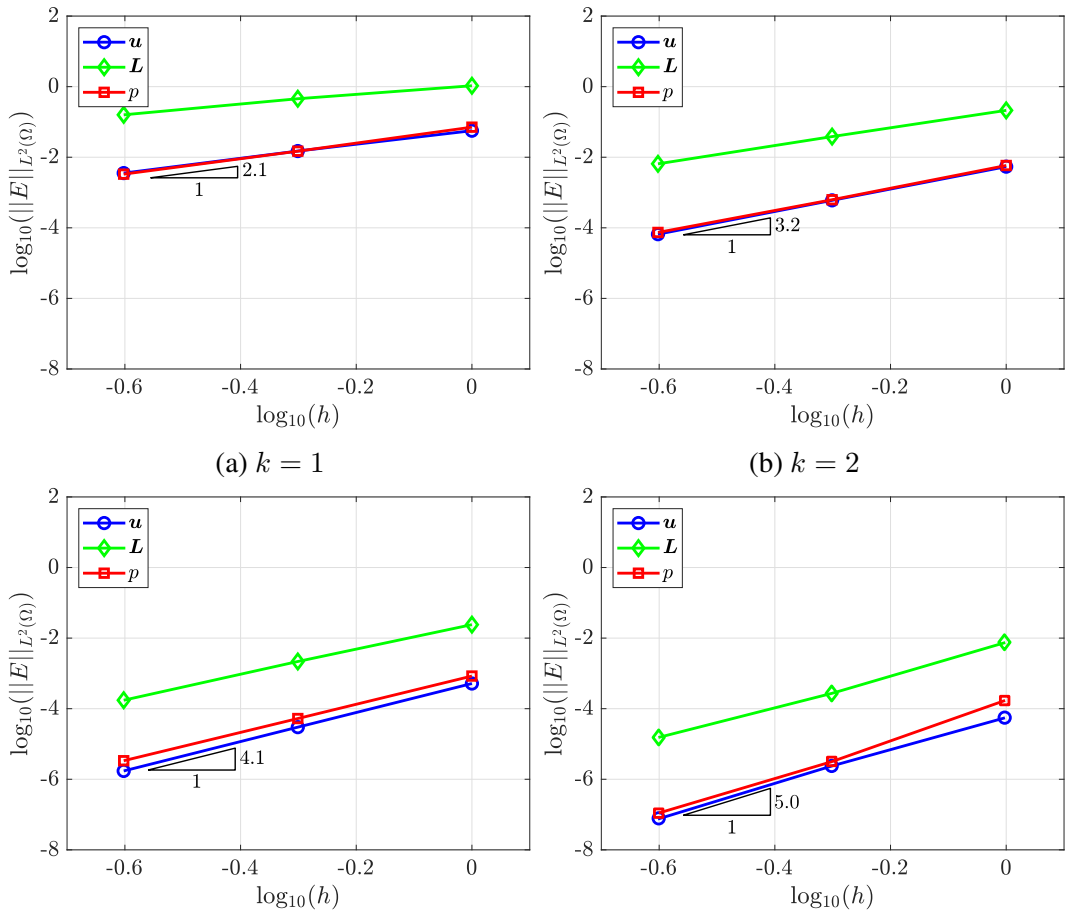
Figure 2.8: Tetrahedral meshes of the domain $\Omega = [0, 1]^3$ used to test the optimal convergence properties of the HDG method.

Figure 2.7 shows the velocity field at $t = 0$.

Three uniform tetrahedral meshes with 24, 192, and 1,536 tetrahedral elements are considered to test the rate of convergence in three dimensions under mesh refinement. The meshes are shown in Figure 2.8.

Figure 2.9 shows the $\mathcal{L}^2(\Omega)$ norm of the error of the velocity, pressure, and mixed variable as a function of the characteristic element size h for a degree of approximation ranging from $k = 1$ up to $k = 4$. To ensure that the error due to the temporal discretisation is lower than the spatial discretisation error, a small enough time step is selected in all the simulations.

As in the two dimensional example, the expected $k+1$ convergence rate can be observed for all the variables and degree of approximation. Compared to the two dimensional



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

Figure 2.9: Ethier-Steinmann Flow: $\mathcal{L}_2(\Omega)$ norm of the error for the velocity, pressure and mixed variable as a function of the characteristic element size h, for different degrees of approximation.

test, it can be observed that the error in the mixed variable is slightly higher than the error of the velocity and pressure. This is mainly due to the higher complexity of the gradient of the velocity in this example.

Similar to the two dimensional example, comparing the results for low and high order approximations provide evidence of the potential of high order approximations. For instance, the solution obtained in the coarsest mesh using $k = 4$, with only 24 elements, is more accurate than the solution obtained in the finest mesh using $k = 1$, with 1,536 elements. Similar to the two dimensional example, it is instructive to consider the size of the global problem as a measure of the cost to compare low and high order approximations. The solution of the problem with $k = 1$ on a mesh of 1,536 elements requires the solution of global problems of 11,136 degrees of freedom, whereas the

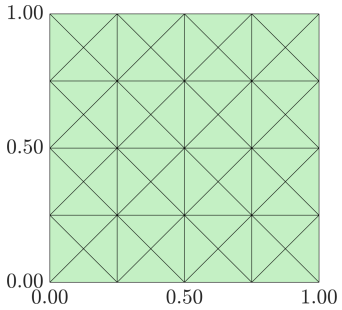


Figure 2.10: Triangular mesh of the domain $\Omega = [0,1]^2$ used to test the optimal temporal convergence properties of the HDG method.

solution of the problem with $k = 4$ on a mesh of 16 elements requires the solution of global problems of 864 degrees of freedom.

2.8.3 Manufactured transient solution

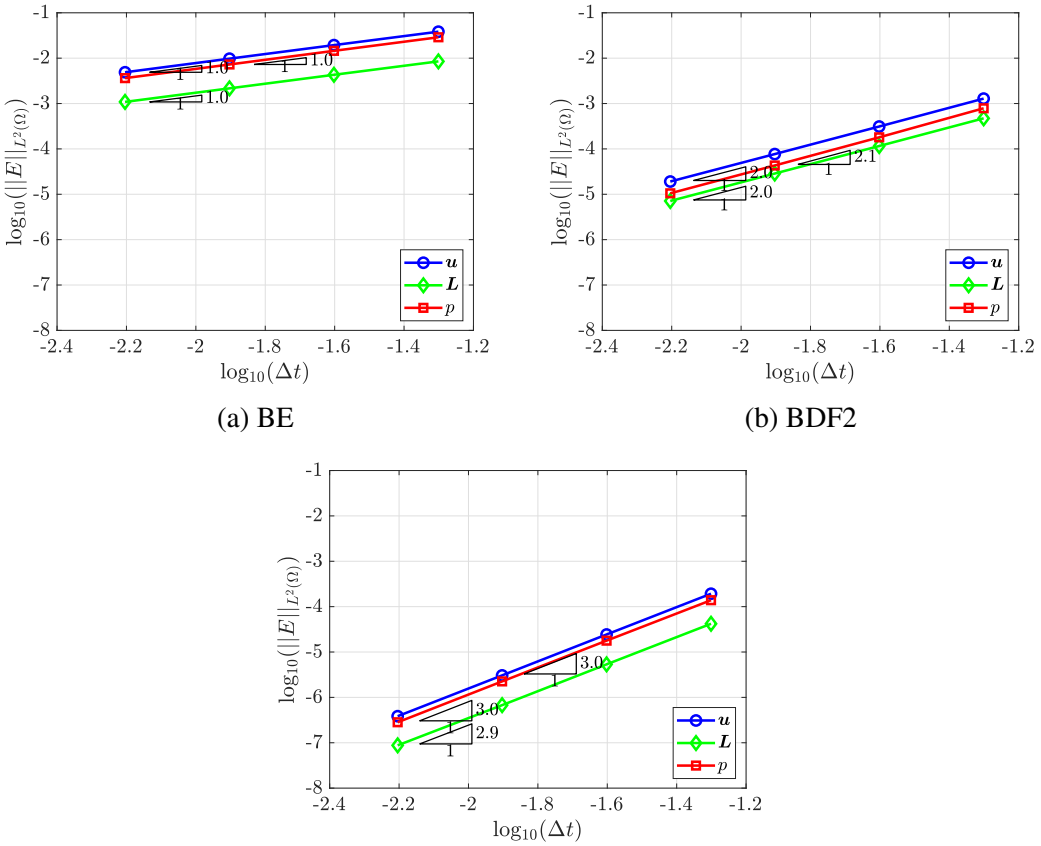
The last example, considered to verify the correct implementation of BDF and ESDIRK time integrators, considers the manufactured solution

$$\boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} \sin(x+\omega t)\sin(y+\omega t) \\ \cos(x+\omega t)\cos(y+\omega t) \end{pmatrix}, \tag{2.60a}$$

$$p(\boldsymbol{x}) = \sin(x-y+\omega t), \tag{2.60b}$$

where the parameter ω is used to adjust rate of change of the velocity and pressure in time. The final time used in these examples is $T_{\texttt{end}} = 0.25$. The mesh used in this example ensures that there is no spatial error that affects the solution, only temporal. The computational domain shown in Figure 2.10 is a square unit $\Omega = [0,1]^2$ and has 64 triangular elements. The degree of approximation is constant in the whole domain and equal to 4.

To test the implementation of the different time integrators considered in this work a fine mesh with a high degree of approximation is considered, so that the error due to the spatial discretisation is negligible when compared to the error due to the temporal discretisation.



(a) BE

(b) BDF2

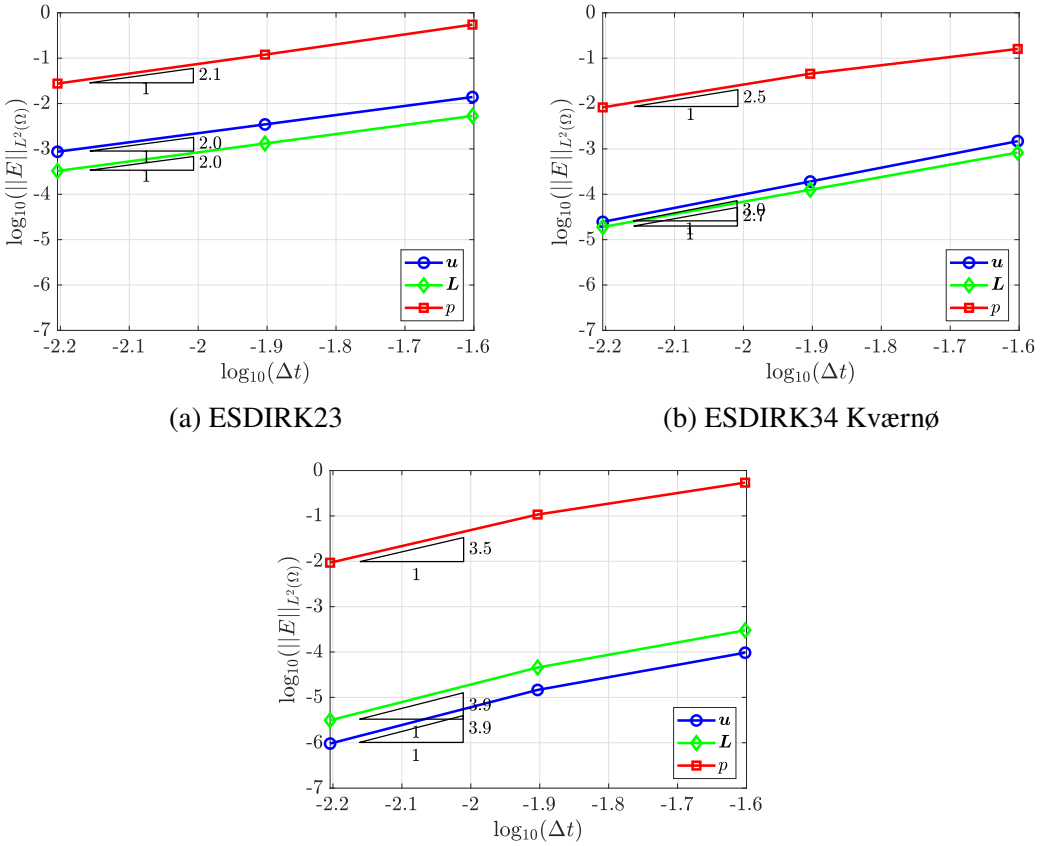
(c) BDF3

Figure 2.11: Manufactured transient solution: $\mathcal{L}_2(\Omega)$ norm of the error for the velocity, pressure and mixed variable as a function of the time step Δt, for different time integration BDF schemes.

Figure 2.11 show the $\mathcal{L}^2(\Omega)$ norm of the error for the velocity, pressure and mixed variable as a function of the time step Δt for the BDF1, BDF2 and BDF3 time integration schemes.

The results show the first, second and third order expected accuracy in all the variables for the BDF1, BDF2 and BDF3 time integration schemes, respectively.

Similar to the superior behaviour of high order approximations in space, the results show the benefit of using high order time integrator. The results with the BDF3 scheme using the coarse time discretisation are more than one order of magnitude more accurate than the results with BDF1 using the finest time discretisation. This means that the results with the BDF3 scheme are more than ten times more accurate than the results with the BDF1 scheme and only using eight time steps.



(a) ESDIRK23

(b) ESDIRK34 Kværnø

(c) ESDIRK46

Figure 2.12: Manufactured transient solution: $\mathcal{L}_2(\Omega)$ norm of the error for the velocity, pressure and mixed variable as a function of the time step Δt, for different time integration ESDIRK schemes.

To test the implementation of the ESDIRK schemes, the value of ω in the analytical solution is increased to 10 to define a much faster variation of the solution in time. This is because for the low value of $\omega = 1$ the highest order ESDIRK methods considered here provide almost machine accuracy and, therefore, do not allow to provide evidence of the optimal convergence rate.

Figure 2.12 presents results for three ESDIRK methods, namely the second-order ESDIRK23 method by (Jørgensen et al., 2018), the third-order ESDIRK34 by (Kværnø, 2004) and the fourth-order ESDIRK46 by (Kennedy and Carpenter, 2016).

The observed convergence rates generally align with the theoretical order of accuracy for each ESDIRK method, despite the fact that some slight loss of accuracy is observed at the pressure, which could be associated with the so-called order reduction of implicit

RK methods (Sanz-Serna et al., 1986) often observed when inhomogeneous boundary conditions are considered.

Chapter 3

A conservative degree adaptive HDG method

3.1 Introduction

The accurate simulation of transient incompressible fluid flows is a central challenge in many CFD applications, including areas such as civil, aerospace, chemical, and biomedical engineering. From a numerical point of view, several difficulties arise when solving the incompressible Navier-Stokes equations due to their non-linear nature and the intricate coupling between velocity and pressure fields (Quartapelle, 2013). When unsteady phenomena are of interest, an extra difficulty is to accurately propagate vortices over long distances.

High-order methods are attractive for the simulation of transient flows due to the lower dissipation and dispersion errors, when compared to low order methods (Ekaterinaris, 2005; Ainsworth et al., 2006; Wang et al., 2013). Continuous and discontinuous Galerkin (DG) methods have their own advantages and disadvantages and have both been successfully applied to a variety of problems in CFD (Chalot and Normand, 2010; Chalot et al., 2015; Gross et al., 2015; Sevilla et al., 2013; Bassi et al., 2007; Liu and Shu, 2000; Montlaur et al., 2010; Ferrer and Willden, 2011; Lehrenfeld and

Schöberl, 2016). Two properties that make DG a preferred option in some cases are the ability to easily handle a variable degree of approximation and the easier definition of the required stabilisation for convection dominated flows (Kompenhans et al., 2016; Ekelschot et al., 2017; Paipuri et al., 2018). The main disadvantage of DG methods is commonly attributed to the duplication of degrees of freedom (Kirby et al., 2012; Yakovlev et al., 2015), a property that facilitates the implementation of variable degree of approximation. Furthermore, as demonstrated by Kirby et al. (2012); Yakovlev et al. (2015), the computational cost of DG methods compared to statically condensed continuous Galerkin methods increases significantly, particularly in three-dimensional problems.

The hybridisable discontinuous Galerkin (HDG) method, originally proposed by Cockburn and co-workers (Cockburn and Gopalakrishnan, 2005, 2009) employs hybridisation to reduce the number of coupled degrees of freedom and has become popular for CFD applications. With HDG, it is possible to use approximations of equal order for both velocity and pressure, circumventing the Ladyzhenskaya-Babuška-Brezzi (LBB) condition. From a computational perspective, the size of the global problem only involves the mean value of the pressure in each element even for high-order approximations, reducing even further the size of the global system of equations to be solved. Furthermore, an important advantage of HDG is the ability to build a super-convergent velocity field (Cockburn and Shi, 2013). The development and application of HDG methods to incompressible flows include the solution of Stokes flows (Cockburn et al., 2010; Nguyen et al., 2010; Cockburn and Shi, 2014, 2013; Giacomini et al., 2018) and the incompressible Navier-Stokes equations (Nguyen et al., 2011; Giorgiani et al., 2014; Rhebergen and Wells, 2018; Gürkan et al., 2019).

To accurately and efficiently capture transient flow phenomena, mesh adaptation techniques are traditionally employed in a low order context. For high-order methods, the use of degree adaptivity offers a new alternative to provide the required accuracy only in the regions of the domain where it is needed, minimising the computational overhead of high-order approximations and circumventing the need to modify the mesh topology.

In the context of HDG, the use of mesh and degree adaptivity has been considered for a variety of problems, including incompressible flows (Giorgiani et al., 2014; Leng, 2021). In HDG methods, the ability to build a super-convergent solution can be used to devise a cheap error indicator to drive the adaptivity. This strategy was first exploited in (Giorgiani et al., 2013) for wave propagation problems.

This chapter first revisits the ability of HDG methods to build a super-convergent velocity field in the context of incompressible flows. Following (Giorgiani et al., 2013, 2014) the super-convergent solution is then used to build a cheap error indicator to drive a degree adaptive process. The implementation of this strategy is demonstrated for a steady problem with known analytical solution. Then, the application of the degree adaptive process is considered for transient problems. First, it is shown that a degree adaptive process can lead to unphysical oscillations in aerodynamic quantities of interest, especially the drag, if the adaptive process reduces the degree of approximation during the time marching process. This phenomenon is related to the violation of the free-divergence condition during the projection of the solution from a space of polynomials of degree r to a space of polynomials of degree s, with $s < r$. Second, this work proposes a conservative projection to guarantee mass conservation during the projection stage. The proposed projection does not introduce significant overhead because it induces the solution of an element-by-element problem and only for those elements where the adaptive process lowers the degree of approximation. Numerical examples are used to illustrate the benefits of the proposed conservative projection using two dimensional examples.

The remainder of this chapter is organised as follows. Section 3.2 presents the strategy to build a super-convergent velocity field. The error indicator that takes advantage of the ability of the HDG method to build a super-convergent velocity is introduced in Section 3.3 and in Section 3.4 the degree adaptation strategy is described. The novel projection scheme to ensure conservation during transient simulations is presented in Section 3.4.2. Section 3.5 presents numerical examples to illustrate the effect of using a standard adaptive process that violates the free-divergence condition during the

projection stage and the benefits of the proposed conservative projection.

3.2 Super-convergent postprocess of the velocity

One of the key advantages of the HDG method is the ability to obtain a super-convergent solution through local postprocessing. This postprocessing technique allows us to compute a new approximation of the velocity field that converges at a higher rate than the original HDG solution, while maintaining important physical properties such as local conservation and H(div)-conformity. H(div)-conformity refers to the property that the normal component of the velocity is continuous across element interfaces. The postprocessing technique is later leveraged to drive an adaptive procedure that locally defines the polynomial degree on each element.

The HDG post-processing procedure requires the solution of a local problem in each element Ω_e to compute an super-convergent velocity approximation $\boldsymbol{u}^\star$. Following the formulation in Giacomini et al. (2020), the postprocessing problem is defined as

$$\begin{cases} \boldsymbol{\nabla} \cdot \left(\sqrt{2\nu}\boldsymbol{\nabla}^{\mathrm{s}}\boldsymbol{u}^\star\right) = -\boldsymbol{\nabla}\cdot\boldsymbol{L}, & \text{in } \Omega_e, \\ \left(\sqrt{2\nu}\boldsymbol{\nabla}^{\mathrm{s}}\boldsymbol{u}^\star\right)\boldsymbol{n} = -\boldsymbol{L}_e\boldsymbol{n}, & \text{on } \partial\Omega_e, \\ (\boldsymbol{u}^\star, 1)_{\Omega_e} = (\boldsymbol{u}, 1)_{\Omega_e}, & \\ (\boldsymbol{\nabla}\times\boldsymbol{u}^\star, 1)_{\Omega_e} = \langle \boldsymbol{u}_D\cdot\boldsymbol{\tau}, 1\rangle_{\partial\Omega_e\cap\Gamma_D} + \langle \hat{\boldsymbol{u}}\cdot\boldsymbol{\tau}, 1\rangle_{\partial\Omega_e\setminus\Gamma_D}, & \end{cases} \tag{3.1}$$

where $\boldsymbol{\tau}$ is the tangential direction to the boundary $\partial\Omega_e$.

The first equation in (3.1) is obtained after applying the divergence operator to the equation that defines the mixed variable, and the boundary condition imposes equilibrated fluxes on the boundary of each element. The two last equations in (3.1) are introduced to remove the indeterminacy associated with the translational and rotational modes.

The weak form of the postprocessing problem is: for $e = 1, \ldots, \mathtt{n_{el}}$, find $\boldsymbol{u}^\star \in$

$[\mathcal{V}_\star^h(\Omega_e)]^{\mathtt{n_{sd}}}$ such that

$$\begin{cases} -(\boldsymbol{\nabla}^s \boldsymbol{v}^\star, 2\nu \boldsymbol{\nabla}^s \boldsymbol{u}^\star)_{\Omega_e} = (\boldsymbol{\nabla}^s \boldsymbol{v}^\star, \boldsymbol{L})_{\Omega_e}, \\ (\boldsymbol{u}^\star, 1)_{\Omega_e} = (\boldsymbol{u}, 1)_{\Omega_e}, \\ (\boldsymbol{\nabla} \times \boldsymbol{u}^\star, 1)_{\Omega_e} = \langle \boldsymbol{u}_D \cdot \boldsymbol{\tau}, 1 \rangle_{\partial\Omega_e \cap \Gamma_D} + \langle \hat{\boldsymbol{u}} \cdot \boldsymbol{\tau}, 1 \rangle_{\partial\Omega_e \setminus \Gamma_D}, \end{cases} \tag{3.2}$$

for all $\boldsymbol{v}^\star \in [\mathcal{V}_\star^h(\Omega_e)]^{\mathtt{n_{sd}}}$, where: $\mathcal{V}_\star^h(\Omega) := \{v \in L_2(\Omega) : v|_{\Omega_e} \in \mathcal{P}^{k+1}(\Omega_e) \forall \Omega_e, e = 1, \ldots, \mathtt{n_{el}}\}$ and $\mathcal{P}^{k+1}(\Omega_e)$ denotes the space of polynomial functions of complete degree at most $k+1$ in Ω_e.

It should be noted that the boundary condition is included in the first equation of (3.2) after the integration by parts.

This postprocessing procedure yields a velocity approximation $\boldsymbol{u}_e^\star$ that is exactly divergence-free, H (div) conforming, and the error converges at a rate of $k+2$ in the $\mathcal{L}^2(\Omega)$ norm for $k \geq 1$, where k is the degree of polynomial used in the original HDG discretisation (Cockburn and Shi, 2013; Giacomini et al., 2020).

Spatial discretisation is performed using isoparametric elements, as done for the other variables and described in Section 2.6. For the postprocessed velocity, polynomials of degree $k+1$ are employed, leading to an element-by-element problem that involves the solution of the linear system of equations.

First, the postprocessed velocity $\boldsymbol{u}^\star$ is approximated using the polynomial shape functions of degree $k+1$, namely

$$\boldsymbol{u}^\star(\boldsymbol{\xi}) \approx \sum_{j=1}^{\mathtt{n_{en}}} \mathbf{u}_j^\star N_j(\boldsymbol{\xi}) \tag{3.3}$$

where $\mathbf{u}_j^\star$ are the nodal values of the postprocessed velocity.

Introducing the approximation of Equation (3.3) into the weak form (3.2) and selecting the space of test functions equations to the space of approximation functions, the

resulting system of equations can be written as

$$\begin{bmatrix} \mathbf{A} & \mathbf{b}_1^T & \mathbf{b}_2^T \\ \mathbf{b}_1 & 0 & 0 \\ \mathbf{b}_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^\star \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ c_1 \\ c_2 \end{bmatrix} \tag{3.4}$$

where

$$A_{ij} = -\int_\Omega 2\nu \nabla N_i \cdot \nabla N_j |\mathbf{J}| d\Omega \tag{3.5}$$

$$f_i = \sum_{j=1}^{\mathtt{n_{en}}} \int_\Omega \nabla N_i \cdot \mathbf{L}_j N_j |\mathbf{J}| d\Omega \tag{3.6}$$

$$(b_1)_i = \int_\Omega N_i |\mathbf{J}| d\Omega \tag{3.7}$$

$$(b_2)_i = \int_\Omega \nabla \times N_i |\mathbf{J}| d\Omega \tag{3.8}$$

$$c_1 = \sum_{j=1}^{\mathtt{n_{en}}} \left(\int_\Omega N_j |\mathbf{J}| d\Omega \right) \mathbf{u}_j \tag{3.9}$$

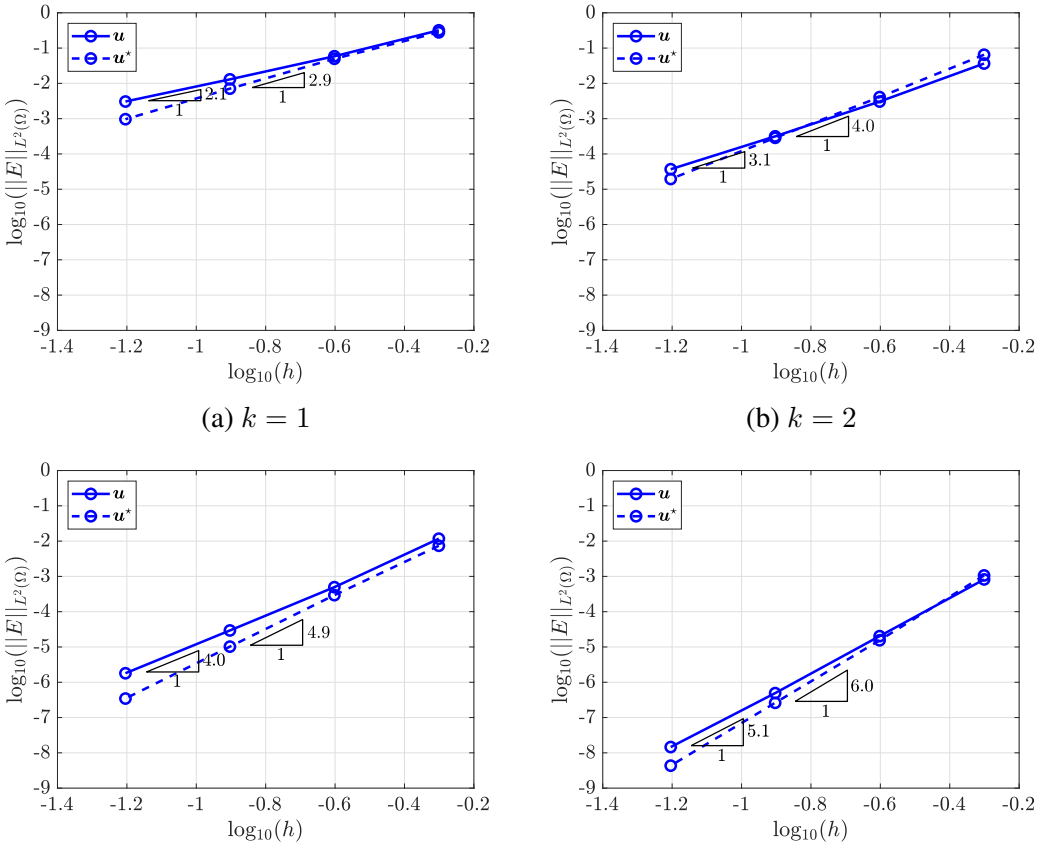
$$c_2 = \sum_{j=1}^{\mathtt{n_{fn}}} \left(\int_{\tilde{\Gamma}} \tilde{N}_j \|\tilde{\mathbf{J}}\| d\tilde{\Gamma} \right) (\boldsymbol{u}_D \cdot \boldsymbol{\tau})_j + \sum_{j=1}^{\mathtt{n_{fn}}} \left(\int_{\tilde{\Gamma}} \tilde{N}_j \|\tilde{\mathbf{J}}\| d\tilde{\Gamma} \right) (\hat{\mathbf{u}} \cdot \boldsymbol{\tau})_j \tag{3.10}$$

λ_1 and λ_2 are Lagrange multipliers used to impose the constraints of Equation (3.2), N_i and $\tilde{N}_j$ are the polynomial shape functions for the element and face, respectively, and $\mathbf{J}$ and $\tilde{\mathbf{J}}$ are the Jacobians for the element and face, respectively.

3.2.1 Verification example

To verify the implementation of the super-convergent postprocess, the Kovasznay flow, previously considered in subsection 2.8.1 is used. The four uniform meshes as shown in Figure 2.5 are considered and the postprocessed velocity is computed for each mesh and for a degree of approximation from $k = 1$ up to $k = 4$.

Figure 3.1 shows the $\mathcal{L}^2(\Omega)$ norm of the error of the velocity and the postprocessed velocity as a function of the characteristic element size. h, for a degree of approximation ranging from $k = 1$ up to $k = 4$.



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

Figure 3.1: Kovasznay Flow: $\mathcal{L}_2(\Omega)$ norm of the error for the velocity, and postprocessed velocity as a function of the characteristic element size h, for different degrees of approximation.

For all polynomial degrees, $\boldsymbol{u}^\star$ consistently shows a higher convergence rate than $\boldsymbol{u}$, and the difference in convergence rates between them is approximately one for all values k. These results indicate that the method achieves optimal convergence for the primary solution and superconvergence for the post-processed solution.

3.3 Error indicator based on the postprocessed velocity

The superconvergent properties of the HDG method allow the construction of an efficient and reliable error indicator based on the post-processed solution. This error indicator, proposed in (Giorgiani et al., 2013) and further used in the context of incompressible flows (Giorgiani et al., 2014; Sevilla and Huerta, 2016), plays a crucial role

in driving adaptive procedures.

Given the HDG solution $\boldsymbol{u}$ and the postprocessed solution $\boldsymbol{u}^\star$, we can define an element wise error indicator E_e as

$$E_e = \left[\frac{1}{|\Omega_e|}\int_{\Omega_e} (\boldsymbol{u}-\boldsymbol{u}^\star)\cdot(\boldsymbol{u}-\boldsymbol{u}^\star)\, d\Omega\right]^{1/2} \tag{3.11}$$

where $|\Omega_e|$ denotes the measure of the element Ω_e, this normalisation becomes crucial when meshes with different element sizes are considered (Díez and Huerta, 1999).

E_e possesses several desirable properties that make it particularly suitable for adaptive procedures in HDG methods. Firstly, it exhibits asymptotic exactness, which means that for smooth solutions, as the mesh size h approaches zero, the error indicator converges to the true error at a higher rate than the HDG solution itself (Giorgiani et al., 2014). This property ensures an increase in the accuracy of the error estimation as the mesh is refined. Secondly, E_e is computationally efficient, as its calculation only involves element-local operations, thus adding minimal overhead to the overall computational cost. Lastly, the error indicator shows high reliability, providing a good estimate of local error in the HDG solution (Nguyen et al., 2010).

3.4 Degree adaptive strategy

Using the error indicator provided in (3.11), this section introduces an automatic degree-adaptive process.

Given a tolerance ε for the velocity error within a region of interest $\Omega_{\text{int}} \subset \Omega$, the adaptive process aims to generate a map of elemental degrees $\{k_{\Omega_e}\}_{e=1}^{\mathtt{n_{el}}}$ such that the error of the velocity field in each element, E_e, is below the required tolerance, namely

$$E_e \leqslant \varepsilon \quad \forall \Omega_e \subset \Omega_{\mathtt{int}}. \tag{3.12}$$

Assuming that the exact solution of the problem is smooth in the sense of having sufficient regularity for the HDG approximation, the error of the velocity field within an element, E_e, behaves asymptotically as

$$E_e \approx C h_e^{\alpha(k_e)}, \tag{3.13}$$

where C is a constant, h_e is the characteristic size of element Ω_e, and α is a constant that depends only on the degree of approximation (k), which depends upon the norm considered.

To achieve a desired error ϵ in a given element, the asymptotic expression of the error can be used, with a different degree of approximation, namely

$$\epsilon \approx C h_e^{\alpha(k_e)+\Delta k_e}, \tag{3.14}$$

where Δk_e is the required change in the degree of approximation to guarantee the desired error.

Taking the logarithm of the ratio of the expressions of ϵ and E_e of equations (3.14) and (3.13) leads to

$$\log_b\left(\frac{\epsilon}{E_e}\right) \approx \Delta k_e \log_b(h_e), \tag{3.15}$$

which provides the required expression for Δk_e, given by

$$\Delta k_e \approx \frac{\log_b(\epsilon/E_e)}{\log(h_e)}. \tag{3.16}$$

In the above expressions, the base b acts as an *aggressiveness parameter* proposed in (Fidkowski and Darmofal, 2007). A lower value of b results in more aggressive refinement, which could lead to larger degree increases but possibly overshooting the optimal degree. In contrast, a larger value of b leads to more conservative refinement, with smaller degree increases, but potentially requiring more adaptive iterations to reach the desired accuracy.

The final expression for the change in the degree of approximation that is used in the current implementation is

$$\Delta k_e = \left\lceil \log_b \left(\frac{\epsilon_e}{E_e} \right) \right\rceil , \tag{3.17}$$

where the value of the tolerance is allowed to be different in each element, ϵ_e, therefore embedding the effect of having elements with very different element sizes. Furthermore, the ceiling function is used to ensure that the resulting value is an integer.

It is important to note that while this method provides a reasonable estimate for degree adaptation, it relies on heuristic principles and may not be optimal in all cases. The effectiveness of this adaptive strategy depends on the accuracy of the error indicator E_e and the appropriate choice of the aggressiveness parameter b and the desired accuracy ϵ_e. These factors should be carefully considered in the context of the specific problem and the desired balance between computational efficiency and solution accuracy. Additionally, the assumption of solution smoothness is crucial, for problems with singularities or sharp gradients, the actual convergence behaviour may deviate from these asymptotic estimates, potentially requiring alternative adaptation strategies.

Furthermore, upper and lower bounds on the polynomial degree are typically set, namely $k_{\text{min}} \leqslant k_{\Omega_i} \leqslant k_{\text{max}}$. The lower bound k_{min} is essential to ensure a proper geometric representation using isoparametric elements, while the upper bound k_{max}, which caps the maximum polynomial degree in the mesh, is imposed for practical reasons.

The strategy used to select these parameters is as follows:

- For the aggressiveness parameter b:
  - Use $b = 10$ for problems with strong local features requiring rapid degree adaptation
  - Use $b = 100$ for smoother problems to avoid oscillatory degree distributions
- For the target accuracy ϵ:
  - Set based on the problem physics and required engineering precision

  - Typical values range from 10^{-3} to 10^{-8}
  - Consider computational cost constraints

- For polynomial degree bounds:
  - Set $k_{\text{min}} \geq 1$ to maintain geometric accuracy
  - Set k_{max} based on available computational resources

The adaptive procedure is applicable to both steady and unsteady problems. In both cases, the process begins with a uniform degree mesh where $k_{\Omega_e} = k_{\text{min}}$ for $e = 1, \ldots, \mathtt{n_{el}}$.

3.4.1 Steady state solutions

For steady state solutions, each iteration of the adaptive process involves computing the steady-state solution to equations (2.51) and (2.53) (neglecting time derivatives), the error (3.11) is evaluated, the degree map is updated using equation (3.17) within the bounds $k_{\text{min}} \leqslant k_{\Omega_i} \leqslant k_{\text{max}}$. The iterative adaptation continues until the prescribed precision ε is achieved throughout the domain of interest, according to Equation (3.12).

A solution derived from a given degree map can serve to enhance the initial guess within the Newton-Raphson scheme through interpolation of the solution at the new nodal distribution within each element. Assume that the solution within an element has been computed using a polynomial approximation of degree r, and the subsequent degree to be used in the element is s. The solution is approximated as

$$\boldsymbol{u}^r(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n}^r_{\mathtt{en}}} \mathbf{u}^r_j N^r_j(\boldsymbol{\xi}), \tag{3.18}$$

where $\mathtt{n}^r_{\mathtt{en}}$ denotes the number of element nodes, $\mathbf{u}_j$ are the nodal values of the solution and N^r_j are the polynomial shape functions of degree r defined, on a reference element,

Algorithm 1 Degree adaptive HDG Method for steady Navier-Stokes Equations

Initialise polynomial degree map $\{k_e\}_{e=1,\dots,\mathtt{n_{el}}}$
Set base b for logarithm and tolerance ε
while true **do**
  for $i_{\mathtt{NR}} \leftarrow 1$ **to** $\mathtt{n_{NR}}$ **do**
    Solve global problem of Equation (2.51) (neglecting time derivatives)
    Solve local problem of Equation (2.53) (neglecting time derivatives)
  end for
  for $i_{\mathtt{el}} \leftarrow 1$ **to** $\mathtt{n_{el}}$ **do**
    Compute super-convergent velocity using Equation (3.2)
    Compute error indicator using Equation (3.11)
    Update the degree using Equation (3.17)
  end for
  if $\max\{E_e : e = 1, \dots, \mathtt{n_{el}}\} \leq \varepsilon$ **then**
    break
  end if
end while

from the set of nodes $\{\boldsymbol{\xi}^r\}_{i=1,\dots,\mathtt{n}^r_{\mathtt{en}}}$. The interpolation onto the novel nodal configuration corresponding to a degree s, $\{\boldsymbol{\xi}^s\}$, can be written as

$$\boldsymbol{u}^s(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n}^s_{\mathtt{en}}} \mathbf{u}^s_j N^s_j(\boldsymbol{\xi}), \tag{3.19}$$

where $\mathbf{u}^s_j = \boldsymbol{u}^r(\boldsymbol{\xi}^s_j)$.

3.4.2 Conservative projection for transient problems

An essential distinction of a degree-adaptive procedure for transient problems, in contrast to the steady-state case, is that the projection of the solution at time t^n onto the desired degree map is required to calculate the solution at time t^{n+1}, rather than merely serving as an initial guess in the Newton-Raphson scheme. Consider a scenario in which the solution in a given element at time t^n is determined with a degree r, and the degree adaptive process subsequently adjusts the required degree within that element to s. The projection methodology, as delineated in (16), generally does not ensure that the projected velocity field at time t^n remains divergence-free. Specifically, if $s \geq r$, that is, if the adaptive process either escalates or maintains the degree of

approximation within an element, the projection does not alter the velocity field at time t^n, given that the polynomial space of degree r is contained within the polynomial space of degree s. Conversely, if $s < r$, that is, if the adaptive process reduces the degree of approximation in the element, the projection modifies the velocity field at time t^n, thus generally breaking the incompressibility constraint.

To avoid this problem, this work proposes a new projection based on the constrained minimisation problem. The objective is to minimise the difference between the initial velocity field ($\boldsymbol{u}^r$) and the corrected velocity field ($\boldsymbol{u}^s$), subject to the mass conservation constraint:

$$\begin{cases} \min_{\boldsymbol{u}_j^s} & E := \int_{\Omega_e} (\boldsymbol{u^s} - \boldsymbol{u}^r) \cdot (\boldsymbol{u^s} - \boldsymbol{u}^r)\, d\Omega \\ \text{s.t.} & \int_{\partial\Omega_e} \boldsymbol{u^s} \cdot \boldsymbol{n} d\Gamma = 0. \end{cases} \tag{3.20}$$

The velocity and the corrected velocity fields are discretised using basis functions of degree r and s respectively

$$\boldsymbol{u}^s \simeq \sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^s} N_i^s \mathbf{u}_i^s, \quad \boldsymbol{u}^r \simeq \sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^r} N_i^r \mathbf{u}_i^r. \tag{3.21}$$

Inserting the polynomial approximation of $\boldsymbol{u}^s$ and $\boldsymbol{u}^r$ into the error function to be minimised, leads to

$$E \simeq \int_{\Omega_e} \left(\sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^s} N_i^s \mathbf{u}_i^s - \sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^r} N_i^r \mathbf{u}_i^r \right) \cdot \left(\sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^s} N_i^s \mathbf{u}_i^s - \sum_{i=1}^{\mathrm{n}_{\mathrm{en}}^r} N_i^r \mathbf{u}_i^r \right) d\Omega \tag{3.22}$$

To find the nodal values of the projected solution, $\mathbf{u}_i^s$, that minimise E the derivative is imposed to vanish, namely

$$\frac{\partial E}{\partial \mathbf{u}_j^s} \simeq 2 \int_{\Omega_e} N_j \left(\sum_i N_i \mathbf{u}_i^s - \sum_i N_i \mathbf{u}_i^r \right) d\Omega = \mathbf{0} \tag{3.23}$$

The discrete version of the constraint in Equation (3.20) is

$$\int_{\partial\Omega_e} \boldsymbol{u}^s \cdot \boldsymbol{n} d\Gamma \simeq \sum_{i=1}^{\mathtt{n}_{\mathtt{en}}^s} \left(\int_{\partial\Omega_e} N_i \boldsymbol{n}, d\Gamma \right) \cdot \mathbf{u}_i^s = 0 \tag{3.24}$$

Combining the optimality conditions and the constraint using a Lagrange multiplier λ, the resulting system of linear equations to be solved in an element where the adaptive process decreases the degree of approximation can be written as

$$\begin{bmatrix} \mathbf{M} & 0 & \mathbf{D}_1 \\ 0 & \mathbf{M} & \mathbf{D}_2 \\ \mathbf{D}_1^T & \mathbf{D}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{U}_1^s \\ \mathbf{U}_2^s \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ 0 \end{Bmatrix} \tag{3.25}$$

in two dimensions, where $\mathbf{U}_a^s$ is the vector containing the nodal values of the a-th component of the projected free-divergence velocity field, λ is the Lagrange multiplier,

$$M_{ij} := \int_{\Omega_e} N_i N_j d\Omega, \quad (D_a)_i := \int_{\partial\Omega_e} N_i n_a d\Gamma, \quad (F_a)_i := \int_{\Omega_e} N_i \mathrm{u}_a^r d\Omega \tag{3.26}$$

and u_a^r is the a-th component of the original velocity field, approximated with polynomials of degree r.

From a computational point of view, the mass matrix $\mathbf{M}$ for elements with an affine mapping linking them to the reference element (e.g. triangular elements with straight edges) can be precomputed in the reference element and scaled using the Jacobian of the isoparametric mapping. For curved elements, the matrix needs to be computed separately for each element. Similarly, the matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ can be precomputed for edges with no variation of the normal (e.g. straight edges in 2D), whereas the computation is performed separately for curved edges.

It is important to underscore that the minimisation problem, specifically the solution of the linear system (3.26), is confined to those elements wherein the adaptive process reduces the degree of approximation. The size of the linear system in two dimensions is $2\mathtt{n}_{\mathtt{en}} + 1$, where $\mathtt{n}_{\mathtt{en}}$ denotes the number of element nodes. Furthermore, since the

problem is resolved independently for each element, it allows for trivial parallelisation, thereby reducing computational overhead.

Algorithm 2 Degree adaptive HDG Method for unsteady Navier-Stokes Equations

Initialise polynomial degree map $\{k_e\}_{e=1,\ldots,\mathtt{n_{el}}}$
Set base b for logarithm, tolerance ε and number of iterations $\mathtt{n_{adaptivity}}$.
for $i_{\mathtt{s}} \leftarrow 1$ **to** $\mathtt{n_{steps}}$ **do**
  for $i_{\mathtt{a}} \leftarrow 1$ **to** $\mathtt{n_{adaptivity}}$ **do**
    for $i_{\mathtt{NR}} \leftarrow 1$ **to** $\mathtt{n_{NR}}$ **do**
      Solve global problem of Equation (2.51)
      Solve local problem of Equation (2.53)
    end for
    for $i_{\mathtt{el}} \leftarrow 1$ **to** $\mathtt{n_{el}}$ **do**
      Compute super-convergent velocity using Equation (3.2)
      Compute error indicator using Equation (3.11)
      Update the degree using Equation (3.17)
      if $\Delta k_e < 0$ **then**
        Compute conservative projection using Equation (3.25)
      end if
    end for
  end for
end for

The algorithm 2 outlines the degree adaptive procedure and the conservative projection proposed, designating $\mathtt{n_{steps}}$ as the total time steps and $\mathrm{n_{NR}}$ as the upper limit of iterations in the Newton-Raphson method.

It should be noted that the methodology described in subsection 3.4.1 is especially beneficial for steady-state problems, as it facilitates multiple iterations to achieve the desired error threshold across the domain. In contrast, for unsteady solutions, the implementation of such an iterative process at each time step like in Alauzet et al. (2007) would be excessively costly. Attempting to achieve the desired error throughout the entire domain at every time step would result in immense computational cost. For time-dependent problems, a more practical approach commonly entails adjusting the solution once before advancing to the next time step, rather than seeking optimal accuracy at each time step. This study proposes a balance between these two methods by iteratively applying the adaptive process for a predetermined number of iterations $\mathtt{n_{adaptivity}}$, typically set at 2 or 3.

3.5 Numerical examples

3.5.1 Wang flow

To assess the performance of the degree adaptive strategy implemented, the so-called Wang flow (Wang, 1991) is first considered. This corresponds to a steady solution of the incompressible Navier-Stokes equations and provides a suitable test because it enables evaluating the accuracy of the error indicator.

The computational domain is defined as $\Omega = [-0.5, 0.5] \times [0, 1]$ and the exact solution is given by

$$\boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} 2ax_2 - b\lambda \exp(-\lambda x_2)\cos(\lambda x_1) \\ b\exp(-\lambda x_2)\sin(\lambda x_1) \end{pmatrix}, \tag{3.27a}$$

$$p(\boldsymbol{x}) = -\frac{1}{2}\exp(2\lambda x_1) + C, \tag{3.27b}$$

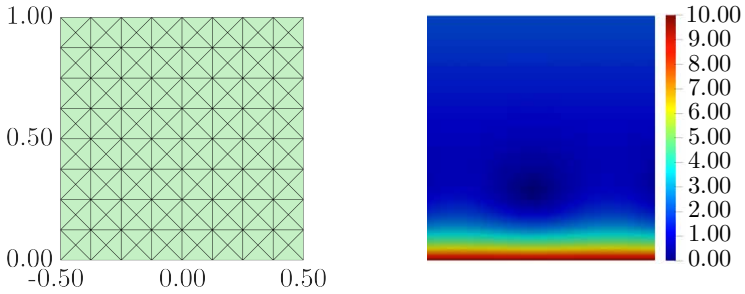
where a, b, and λ are parameters, selected as $a = 1$, $b = 1$, and $\lambda = 10$, following (Giorgiani et al., 2014) and C is a constant chosen to ensure zero mean pressure in the domain. The chosen parameters lead to a velocity field with strong gradients near the bottom boundary, providing a suitable setting to test the degree adaptive procedure.

Dirichlet boundary conditions, corresponding to the exact velocity, are imposed on all boundaries except the bottom part, where a Neumann condition, also corresponding to the exact solution, is imposed.

The coarse uniform triangular mesh shown in Figure 3.2a is considered

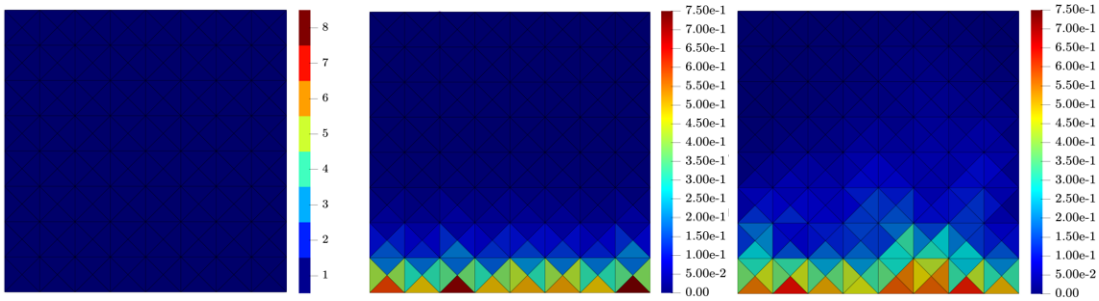
To illustrate the adaptive process, a tolerance of $\epsilon = 10^{-8}$ is used for all elements and the aggressiveness parameter is selected as $b = 100$.

Figure 3.3 shows the initial degree map, which is taken as $k = 1$ for all the elements together with the error indicator and the exact error. Given the coarse mesh employed



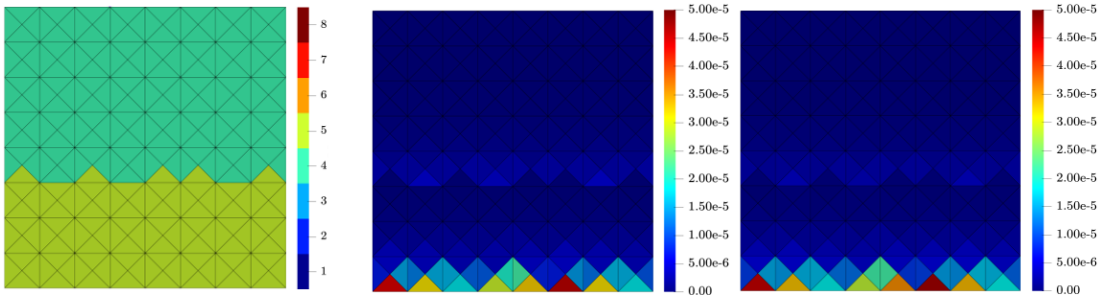
(a) Unstructured triangular mesh (b) Analytical velocity field

Figure 3.2: Wang flow: problem setup.



(a) Degree map (b) Error indicator (c) Exact error

Figure 3.3: Wang flow: Initial computation with linear approximation in all elements.

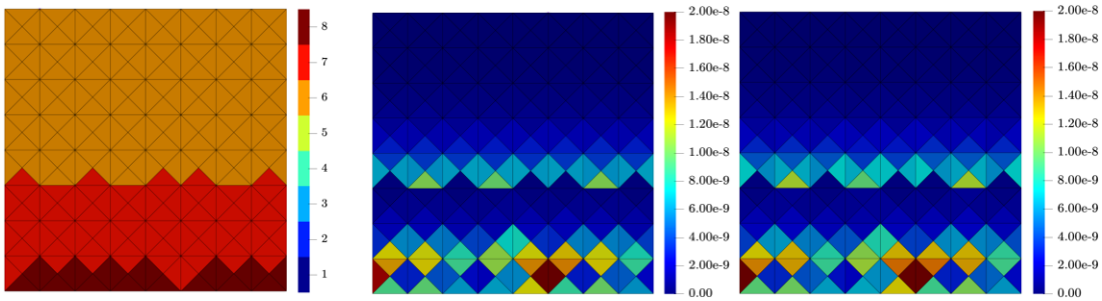


(a) Degree map (b) Error indicator (c) Exact error

Figure 3.4: Wang flow: First iteration of the degree adaptive process.

and the low order approximation the error is obviously high, but, more importantly the error indicator maps resembles the exact error, providing evidence of the suitability of the error indicator to drive a degree adaptive process.

Using the error indicator obtained from the first computation, the degree map, shown in Figure 3.4, is obtained following the strategy presented in section 3.4. Figure 3.4 also shows the error indicator and the exact error. The results show a sudden increase of the order of approximation, resulting in elements near the bottom boundary with $k = 5$ and $k = 4$ for the rest of the domain. The maximum elemental error is approximately



(a) Degree map (b) Error indicator (c) Exact error

Figure 3.5: Wang flow: Second iteration of the degree adaptive process.

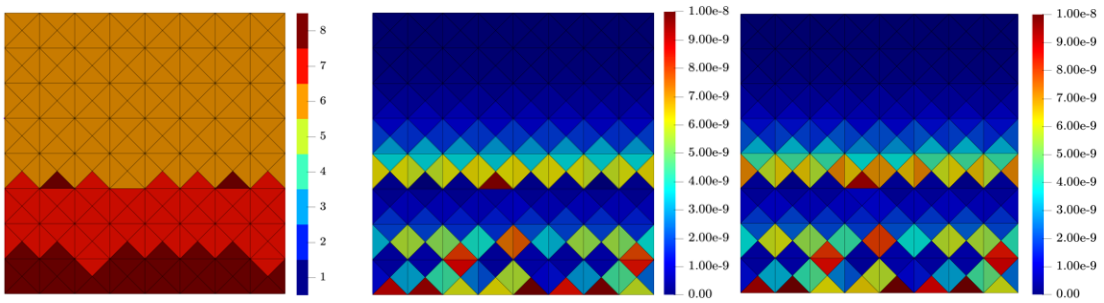
5×10^{-5}, which is four orders of magnitude lower than in the initial computation. Some localised regions with high errors persist, meaning that more iterations are required to guarantee that the error is below the desired tolerance.

Figures 3.5 and 3.6 show the next two iterations of the degree adaptive process. The results show not only the ability of high order approximations to provide extremely low errors, i.e. of the order of 10^{-8}, but, more importantly, the quality of the error indicator is demonstrated by its efficiency index η, defined as the ratio between the estimated and exact errors minus one. Mathematically, this is expressed as:

$$\eta = \frac{E_e}{E_e^{\text{exact}}} - 1 \tag{3.28}$$

where E_e is the error indicator and E_e^{exact} is the exact error computed using the analytical solution. The error indicator demonstrates excellent performance, with efficiency values varying between -0.035 and -0.005, indicating that the error indicator very closely approximates the true error. This high efficiency is achieved because of the superconvergent properties of the HDG method and enables reliable error indicator for adaptive procedures.

In the final iterationFigure 3.6 presented, the k map (a) shows a sophisticated distribution of polynomial degrees, with the highest orders concentrated near the bottom and in specific regions throughout the domain. Both error maps (b, c) demonstrate a more uniform and significantly reduced error distribution across the entire domain.



(a) Degree map (b) Error indicator (c) Exact error

Figure 3.6: Wang flow: Third iteration of the degree adaptive process.

3.5.2 Flow around two circular cylinders

The second example considers laminar flow, at $Re = 100$, around two circular cylinders in tandem. This example is used to study the importance of the proposed projection when performing a degree adaptive computation of transient flows.

The far field is made of a circle of diameter 100 centred at the origin, whereas the two circular cylinders have diameter 1 and are centred at $(-20, 0)$ and $(10, 0)$, respectively.

An unstructured mesh of 2,712 triangles is used for this example shown in Figure 3.7. Curved elements are generated near the cylinder using the elastic analogy presented in (Xie et al., 2013). Given the low Reynolds number considered, the size of the elements in the normal direction to the wall is relatively large, and only the first two layers of elements around the cylinders are curved. More precisely, the size of the first element around the circular cylinders is 0.01 and the growth factor in the normal direction is 1.4. Two point sources are introduced to prescribe a mesh size of 0.2 near the cylinders, whereas a line source with size 0.75 is placed in the path of the von Karman vortex street. A detailed view of the mesh near the cylinders is shown in Figure 3.8.

The ESDIRK46 time marching algorithm (Kennedy and Carpenter, 2016) is used with a time step $\Delta t = 0.2$ and the solution is advanced until the final time $T = 200$.

Since an analytical solution for this problem is not available, a reference solution is

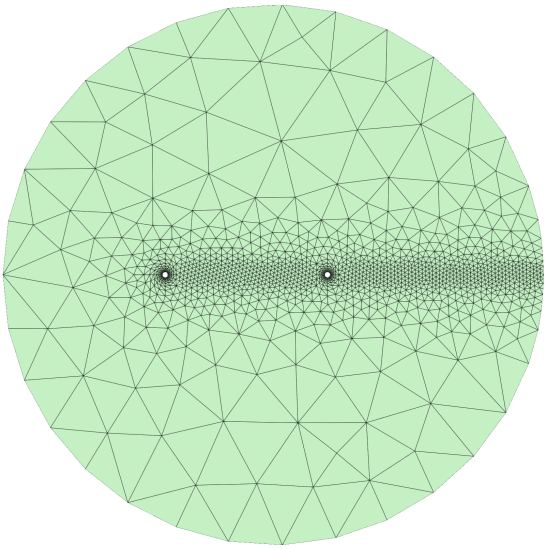


Figure 3.7: Flow around two circular cylinders: Unstructured triangular mesh of the whole domain.

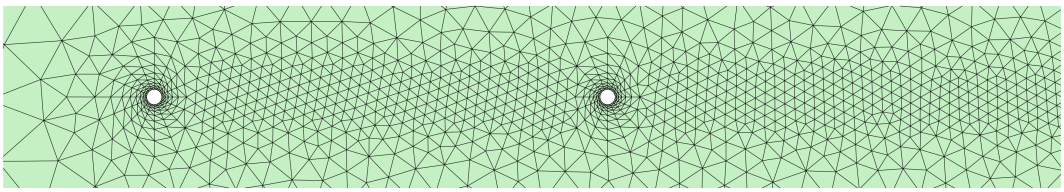
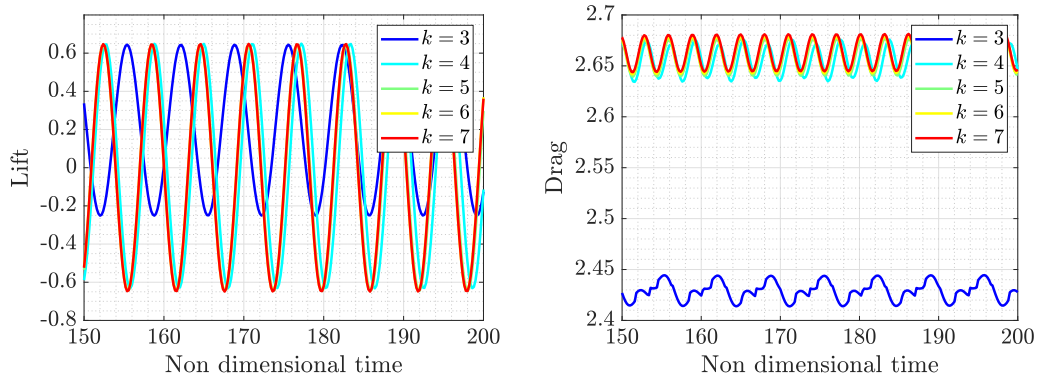


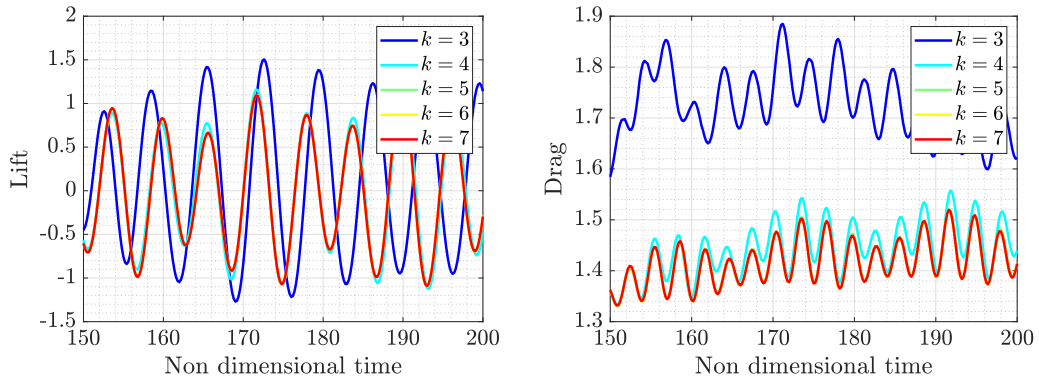
Figure 3.8: Flow around two circular cylinders: detail of the unstructured triangular mesh near the circular cylinders.



(a) Lift (b) Drag

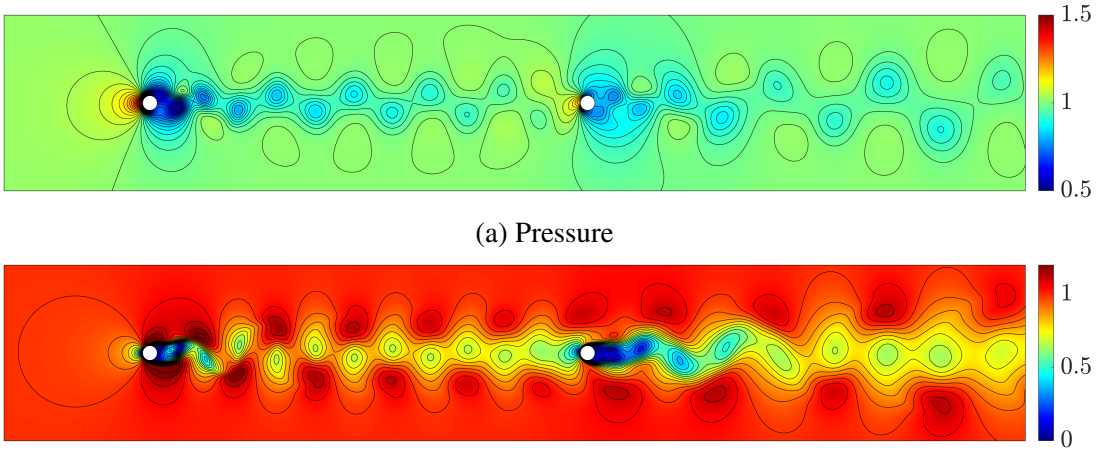
Figure 3.9: Flow around two circular cylinders: lift and drag over the first cylinder using uniform degree across the domain.

established through a convergence test by increasing the uniform degree of approximation, from $k = 3$ to $k = 7$. Figure 3.9 shows the lift and drag for the first cylinder as a function of time and for different degrees of approximation. Similarly, Figure 3.10 shows the lift and drag for the second cylinder. The results show that employing a



(a) Lift (b) Drag

Figure 3.10: Flow around two circular cylinders: lift and drag over the second cylinder using uniform degree across the domain.



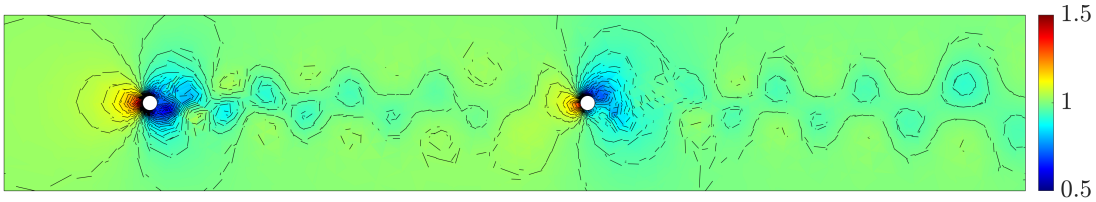
(a) Pressure

(b) Velocity

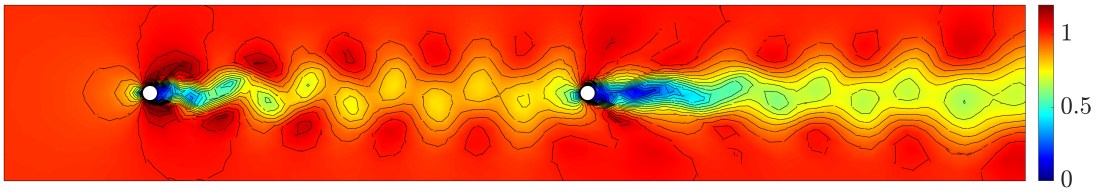
Figure 3.11: Flow around two circular cylinders: Pressure and magnitude of the velocity fields at $t = 200$ with a uniform degree of approximation $k = 6$.

uniform degree of approximation $k = 6$ is enough to obtained a converged solution, and this is considered as the reference solution in all the experiments reported here. Figure 3.11 shows the reference pressure and magnitude of the velocity at $t = 200$.

In this problem, a high-order spatial approximation is essential to accurately depict the von Karman vortex street produced by the first cylinder and its effect on the second cylinder. When using a first-order ($k = 1$) approximation on the same mesh, the vortex intensity is missed, as demonstrated in Figure 3.12, clearly showcasing the low-dissipation characteristics of a high-order approximation scheme. The low order results also display a larger dispersion when compared to the high order approximation as the vortices appear in different positions.



(a) Pressure

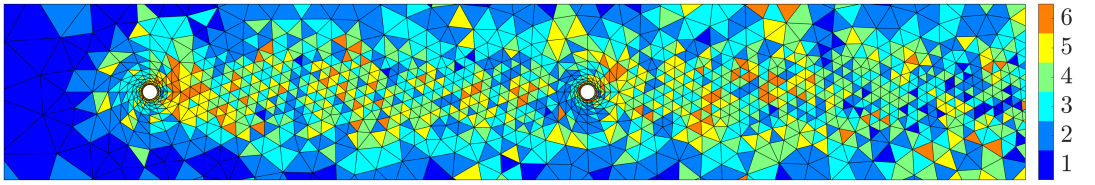


(b) Velocity

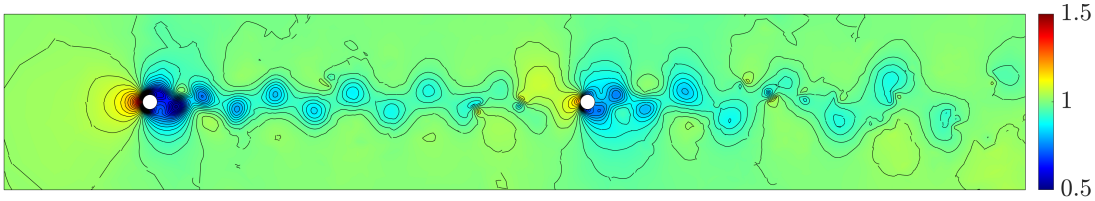
Figure 3.12: Flow around two circular cylinders: Pressure and magnitude of the velocity fields at $t = 200$ with a uniform degree of approximation $k = 1$.

The results illustrated in Figure 3.11 indicate that a consistent degree of approximation is unnecessary and that adopting a degree-adaptive strategy is beneficial to improve resolution where necessary. Subsequent experiments evaluate various degree-adaptive methodologies, with a target error of $\varepsilon = 10^{-4}$ as specified in Section (3.3), unless otherwise noted. Due to the large time step used, the adaptive procedure is executed twice per time step to accurately capture flow characteristics as the solution advances. This example also demonstrates the effects of not repeating the adaptive process.

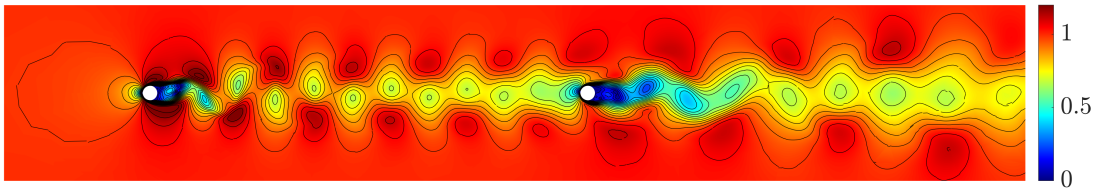
A typical degree-adaptive method is initially examined, omitting the suggested correction. For each time step, the solution within each element is projected using the target degree of approximation map based on the error indicator supplied by the HDG method. Figure 3.13 illustrates the results at $t = 200$, along with the degree used for each element. The velocity field aligns well with the reference solution, with only a slight reduction in vortex intensity behind the circular cylinders. However, the pressure field reveals significant numerical artefacts compared to the reference solution. These artefacts arise from violating the incompressibility constraint when transferring the velocity field between different degree maps, particularly when the approximation degree is lowered, as detailed in Section 3.4.2. To evaluate the precision of the simulations, the quantities of interest are the lift and drag. Figure 3.14 presents the lift and drag in the first cylinder using a standard degree adaptive approach, without the application of



(a) Degree map

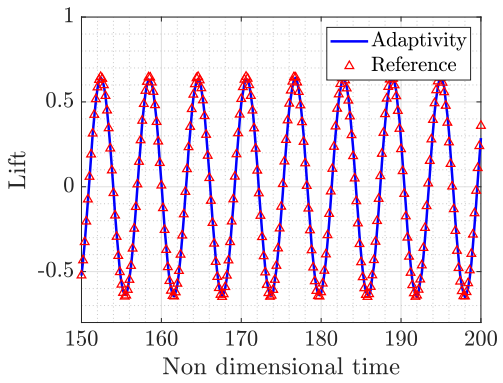


(b) Pressure

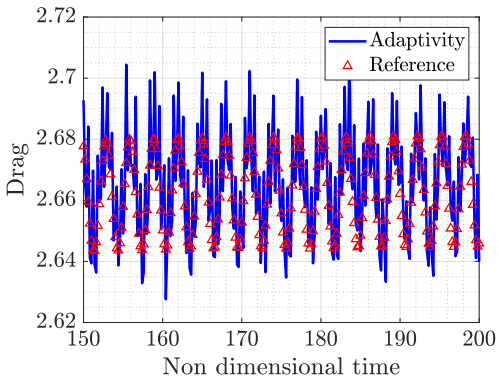


(c) Velocity

Figure 3.13: Flow around two circular cylinders: Pressure and magnitude of the velocity fields at $t = 200$ with degree adaptivity.



(a) Lift

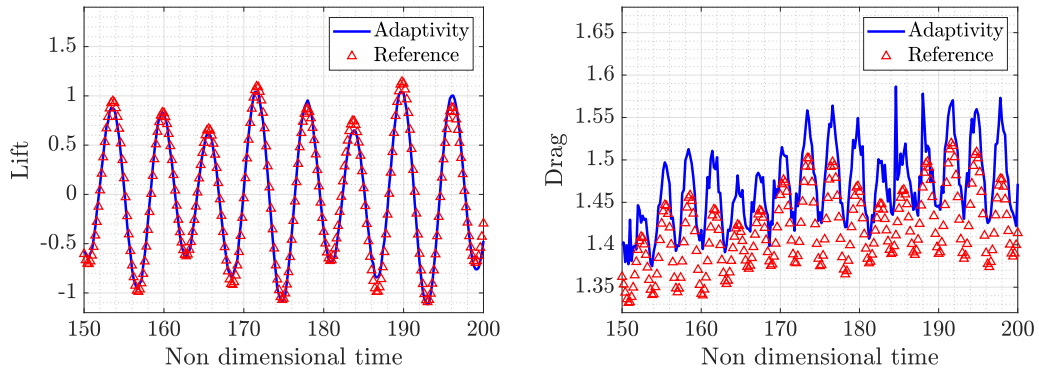


(b) Drag

Figure 3.14: Flow around two circular cylinders: lift and drag over the first cylinder using degree adaptivity compared to the reference solution.

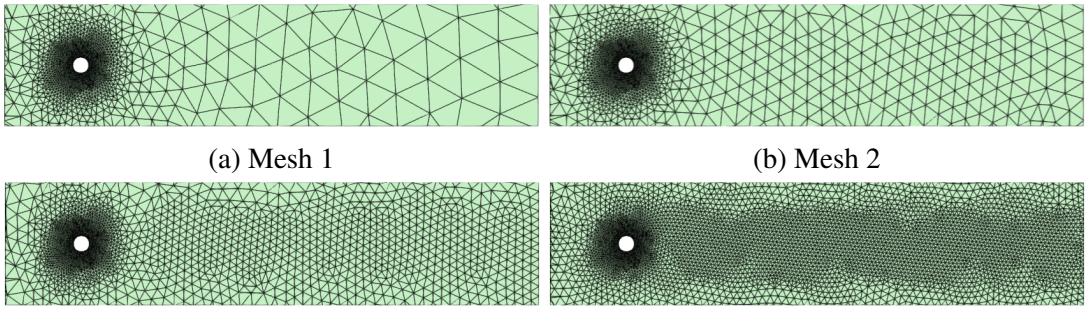
the proposed correction, and the results are juxtaposed with the reference solution.

The outcomes distinctly reveal non-physical oscillations in the drag, while the lift is computed precisely. Comparable results for the relevant quantities related to the second cylinder are illustrated in Figure 3.15. To examine the observed greater accuracy in lift measurements, a mesh convergence study was performed for steady flow around a cylinder at a Reynolds number $Re = 30$. Four different meshes, illustrated in



(a) Lift (b) Drag

Figure 3.15: Flow around two circular cylinders: lift and drag over the second cylinder using degree adaptivity compared to the reference solution.



(a) Mesh 1 (b) Mesh 2

(c) Mesh 3 (d) Mesh 4

Figure 3.16: Flow around a circular cylinder: Sequence of refined triangular meshes used to test the convergence of lift and drag force calculations.

Figure 3.16, were used, containing 1058, 1148, 1506, and 2922 triangular elements, respectively. The study examined lift and drag forces on the upper and lower sections of the cylinder separately. Table 3.1 and Table 3.2 present the results of this analysis.

The data reveal that as mesh refinement increases, both lift and drag values for the upper and lower cylinder sections converge toward reference values. However, when evaluating the error in total lift and drag, a discrepancy appears. The total drag exhibits the expected error reduction with mesh refinement, whereas the total lift demonstrates remarkably small errors even on coarser meshes. This phenomenon can be attributed to error cancellation and the symmetry of the lift force about a zero mean value. The lift forces on the upper and lower sections of the cylinder have opposite signs, leading to a significant reduction in error when summed to calculate the total lift. In contrast, drag forces act in the same direction in both sections, preventing such error cancellation.

Mesh	Upper half		Bottom half		Total	
	Value	**Error**	**Value**	**Error**	**Value**	**Error** $[10^{-5}]$
1	-2.4893949	0.0143736	2.4896956	0.0140012	0.0005007	2.76
2	-2.4989936	0.0048749	2.4993277	0.0049691	0.0004341	9.42
3	-2.5036432	0.0001253	2.5042879	0.0000089	0.0006447	11.64
4	-2.5037685	-	2.5042968	-	0.0005283	-

Table 3.1: Steady flow around a cylinder at $Re = 30$: Lift values and corresponding errors for different meshes.

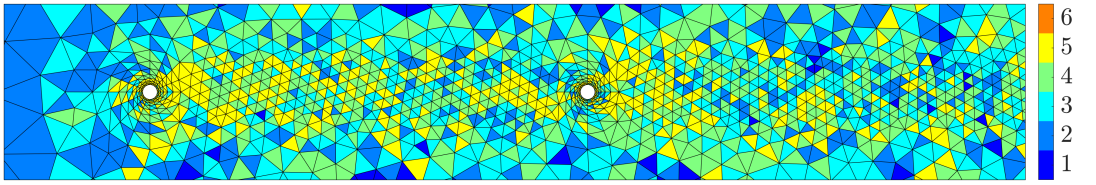
Mesh	Upper half		Bottom half		Total	
	Value	**Error**	**Value**	**Error**	**Value**	**Error** $[10^{-3}]$
1	1.7486673	0.0063593	1.7479117	0.0063767	3.4967790	12.736
2	1.7452090	0.0027010	1.7441625	0.0026275	3.4893715	5.3285
3	1.7422329	0.0002751	1.7414490	0.0000860	3.4836819	0.3611
4	1.7425080	-	1.7415350	-	3.4840430	-

Table 3.2: Steady flow around a cylinder at $Re = 30$: Drag values and corresponding errors for different meshes.

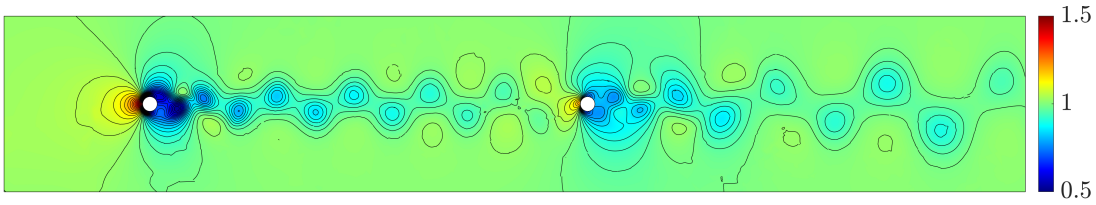
The degree-adaptive method is further improved by incorporating the correction proposed in Section 3.4.2. To demonstrate the advantages of this refined approach, Figure 3.17 depicts the degree map, the pressure distribution, and the velocity magnitude at $t = 200$. The lack of artefacts in the pressure field is remarkable, and there is significant agreement with the reference solution.

To more accurately measure the precision of the simulation using the suggested conservative projection, Figure 3.18 presents the lift and drag forces on the first cylinder. The results demonstrate that the suggested correction effectively removes the non-physical oscillations seen in previous simulations and produces lift and drag measurements closely matching the reference solution. Figure 3.19 shows the results for the second cylinder, again verifying the absence of oscillations and the strong agreement with the reference solution.

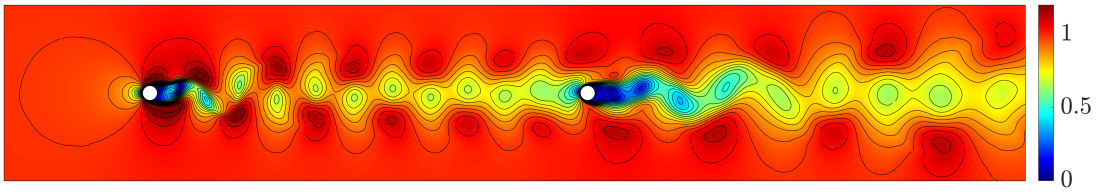
To better demonstrate the advantage of the suggested conservative projection, Table 3.3 presents the maximum error in the lift and drag forces for both cylinders. The results



(a) Degree map

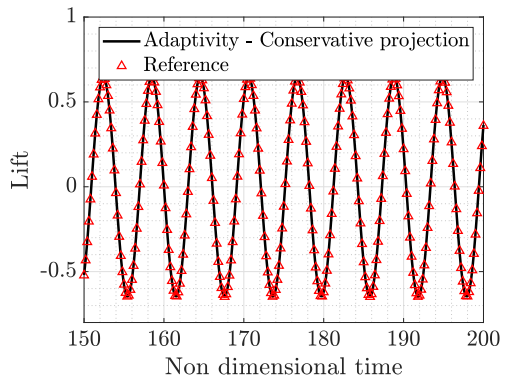


(b) Pressure

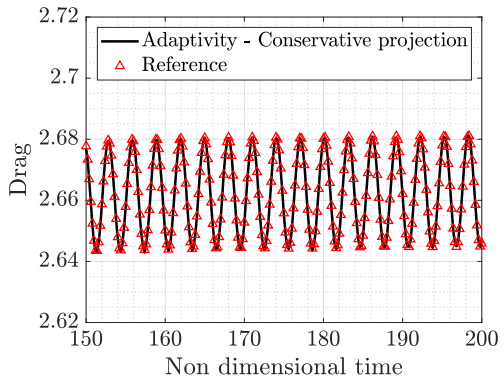


(c) Velocity

Figure 3.17: Flow around two circular cylinders: Pressure and magnitude of the velocity fields at $t = 200$ with degree adaptivity and the conservative projection.



(a) Lift

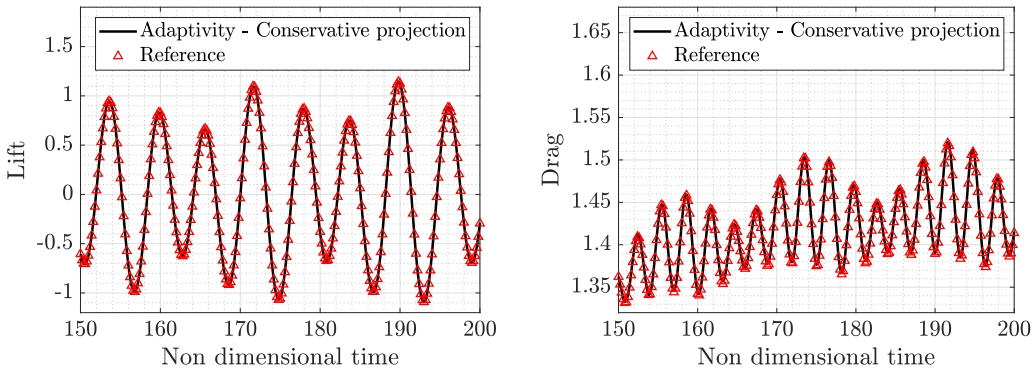


(b) Drag

Figure 3.18: Flow around two circular cylinders: lift and drag over the first cylinder using degree adaptivity and the proposed correction compared to the reference solution.

clearly demonstrate the enhanced accuracy delivered by the conservative projection. Specifically, the error in the lift force is reduced by an order of magnitude, while the error in the drag force is reduced by nearly 40 times when the conservative projection is employed.

To conclude, additional numerical experiments are conducted to demonstrate that conservative projection becomes necessary only when the degree of approximation is



(a) Lift (b) Drag

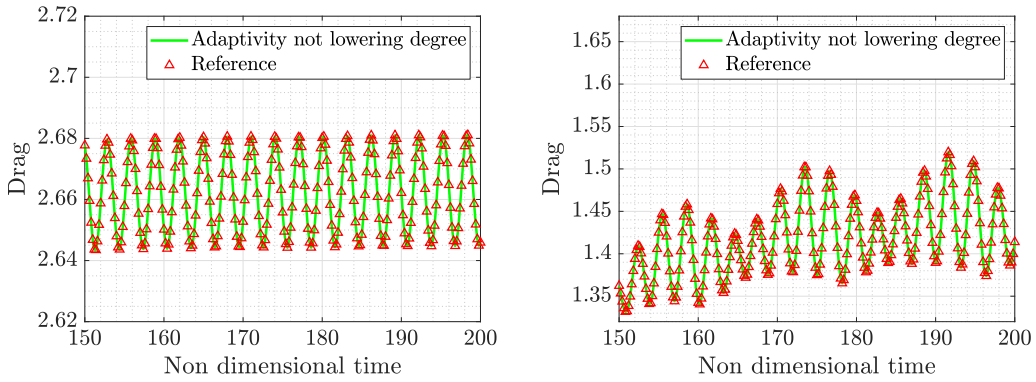
Figure 3.19: Flow around two circular cylinders: lift and drag over the second cylinder using degree adaptivity and the proposed correction compared to the reference solution.

	Cylinder 1		**Cylinder 2**	
	Standard adaptivity	**Conservative projection**	**Standard adaptivity**	**Conservative projection**
Lift error	8.1×10^{-2}	6.8×10^{-3}	1.8×10^{-1}	1.5×10^{-2}
Drag error	3.7×10^{-2}	1.0×10^{-3}	1.8×10^{-1}	4.6×10^{-3}

Table 3.3: Flow around two circular cylinders: maximum error in lift and drag for the two cylinders using the standard adaptivity and the adaptivity with the proposed conservative projection.

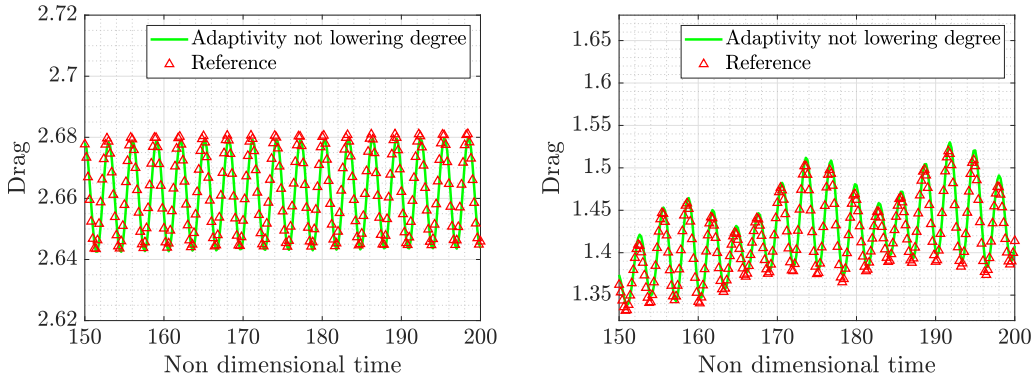
allowed to decrease during the adaptive process. Moreover, the influence of the target error on the degree-adaptive process is exemplified.

Figure 3.20 shows the drag on the first and second cylinders using a standard degree adaptivity where the degree of approximation is not allowed to decrease. The reference solution aligns closely without exhibiting the oscillatory behavior seen when the degree was reduced during the adaptive process. However, a significant disadvantage of this method is its increased computational cost. If an element achieves a high degree of approximation at one time step, it retains this degree for the duration of the simulation, even when it is unnecessary for feature capture in that region going forward. In this scenario, due to the impulsive start and the stringent error requirement per element $\varepsilon = 10^{-4}$, every mesh element demands, at some point, an approximation degree $k = 6$. Thus, this method matches the reference solution, but with the added expense of computing the error indicator and projecting the solution at each time step. Using



(a) Lift (b) Drag

Figure 3.20: Flow around two circular cylinders: Drag on the two cylinders using degree adaptivity and not allowing the degree to be decreased during the adaptive process.



(a) Lift (b) Drag

Figure 3.21: Flow around two circular cylinders: Drag on the two cylinders using degree adaptivity and not allowing the degree to be decreased during the adaptive process with $\varepsilon = 10^{-3}$.

a more relaxed tolerance in the adaptive process, specifically $\varepsilon = 10^{-3}$, yields the quantities of interest without oscillations, as demonstrated in Figure 3.21, suggesting that the root cause of drag oscillations is the breach of the incompressibility condition during the projection of the solution to a lower degree. Some discrepancies in the drag of the second cylinder are visually observed due to the use of a less restrictive tolerance.

The degree map at $t = 200$ when the adaptive process is implemented without allowing the degree of approximation to be decreased and with $\varepsilon = 10^{-3}$ is shown in Figure 3.22. Comparing the degree map of the adaptivity process with the suggested correction, as illustrated in Figure 3.17, it is evident that most elements in the wake of the two cylinders

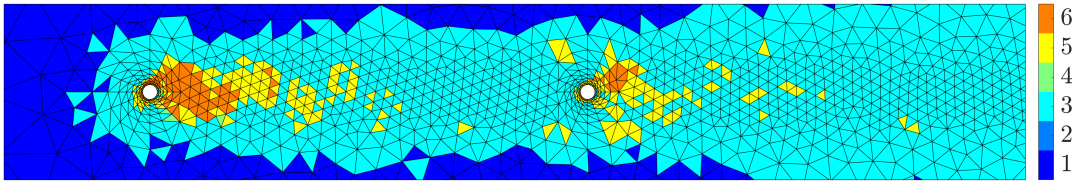


Figure 3.22: Flow around two circular cylinders: Degree of approximation at $t = 200$ not allowing the degree to be decreased during the adaptive process.

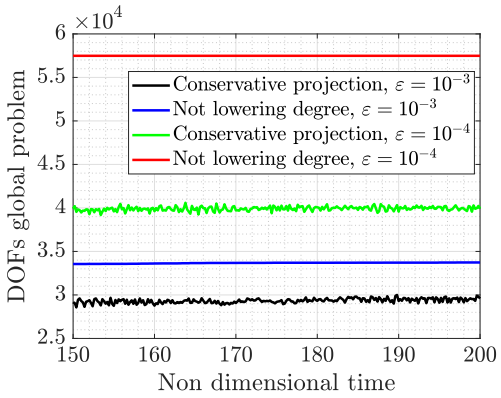


Figure 3.23: Flow around two circular cylinders: Number of degrees of freedom of the global problem for two different adaptive approaches and for two different values of the desired error.

are maintained at a higher degree when the adaptivity process is restricted from lowering the degree. Furthermore, it can be observed that when degree reduction is prohibited, several elements in the wake of the two cylinders continue to use an approximation degree of $k = 6$. In contrast, if the adaptivity process allows degree reduction, this high degree of approximation is unnecessary at the final stage. Figure 3.23 demonstrates how the adaptive process decreases the degree of freedom by allowing a reduction in the degree of approximation. The findings clearly highlight the benefits of employing the proposed projection for adjusting the degree during the time stepping process. Importantly, the lower the target error, the more beneficial it is to allow a reduction in degree.

Regarding computational cost, the simulation with the proposed conservative projection is nearly twice as fast as the simulation with a uniform degree of approximation of $k = 6$. Furthermore, the simulation using the conservative projection is more than three times faster than the simulation without degree reduction. In fact, simulations

Type of core	CPU
Number of cores	1
CPU Model	Intel® Xeon® Gold 6252
Memory available	96GB-192GB
Platform	Local server
Geographical location	United Kingdom
Real CPU usage factor	1.0
Power usage efficiency (PUE)	1.4

Table 3.4: Details of the computational infrastructure used.

without reducing the degree are more costly than computing the reference solution because most elements end up with the maximum degree of approximation, and the expense of computing the error indicator and projecting the solution twice per time step becomes significant. This shows that reducing the degrees of freedom leads to a substantial decrease in computational time.

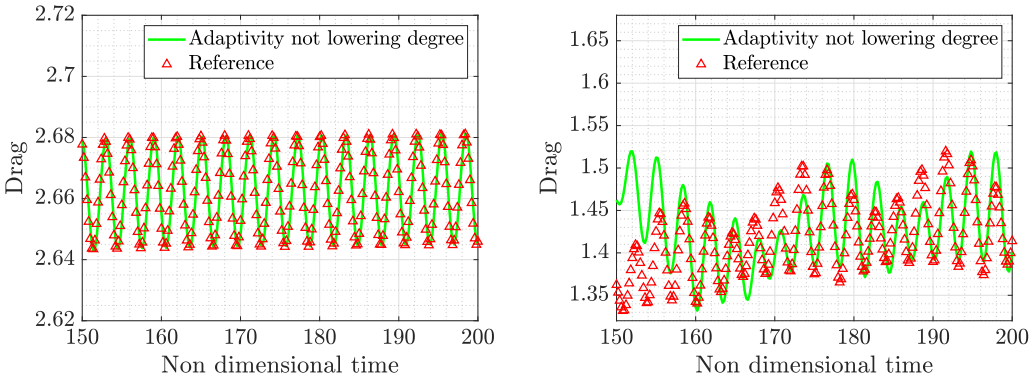
All numerical experiments were performed using a single core on a local server equipped with an Intel® Xeon® Gold 6252 processor and 96-192GB of available memory. The linear systems arising from the HDG discretization were solved using the parallel direct solver PARDISO from the Intel MKL library. Table 3.4 provides details of the computational infrastructure used.

To assess the computational efficiency of the proposed methodology, Table 3.5 presents a detailed comparison of the computational costs for different solution strategies applied to the flow around two circular cylinders. The results demonstrate that the conservative projection approach achieves significant computational savings compared to both uniform approximation and standard adaptivity, while maintaining solution accuracy.

Lastly, a numerical experiment illustrates the necessity of performing the adaptive process twice per time step. The simulation depicted in Figure 3.21 is repeated, but with degree adaptivity performed only once per time step. Due to the large time step used with a high-order time integrator, the computed drag shows a notable loss in accuracy, as shown in Figure 3.24.

Method	**Total walltime** (h)	**Error indicator** (%)	**Projection time** (%)	**# DOFs** (avg)
Uniform (k=6)	19.42	-	-	60,196
Standard adaptivity	5.68	8.3	5.2	39,450
Conservative projection	10.58	8.5	6.8	29,750
No degree reduction	25.42	8.1	5.5	57,200

Table 3.5: Flow around two circular cylinders: Computational costs for simulation up to T=200.



(a) Lift (b) Drag

Figure 3.24: Flow around two circular cylinders: Drag on the two cylinders using degree adaptivity, not allowing the degree to decrease during the adaptive process with $\varepsilon = 10^{-3}$ and performing the adaptivity only once per time step.

Chapter 4

Neural network-driven degree adaptivity

4.1 Introduction

The use of Artificial Neural Networks (ANNs) in CFD applications has attracted great attention in recent years due to the possibility to improve the efficiency and accuracy of numerical simulations (Brunton et al., 2020). This chapter introduces an innovative methodology that employs ANNs to predict flow solutions at a future time and its application to drive a degree process in transient incompressible viscous flow simulations. The objective is to accelerate parametric analysis, where simulations need to be repeated multiple times for varying parameters that involve different flow conditions. The application example to be used in the next chapter involves the simulation of gust where the same simulation is to be repeated multiple times for different gust conditions such as amplitude, width or angle of the gust.

Traditional adaptive methods typically solve the governing equations at a given time instant before moving on to the next instant (Kompenhans et al., 2016; Giorgiani et al., 2013, 2014; Ekelschot et al., 2017). As shown in the numerical examples, flow features might be lost due to the fact that the adaptation is performed before advancing the

solution in time. An obvious solution consists of repeating each time step to ensure that the mesh has enough resolution where is needed. However, this approach is computationally expensive, as it basically doubles the cost of a single CFD simulation. The proposed neural network-driven method aims to predict the solution before the time step is performed, allowing for adaptation to take place in areas where it will be needed, ultimately providing a more efficient use of computational resources.

The main idea is to train an ANN from available simulations to learn the complex relationship between the solution at a given time step and the solution at the next time step. To use a simple multilayer perceptron, which is usually easy to train compared to other types of networks, it is necessary to define an architecture with a fixed number of inputs and outputs. Given that the meshes considered in CFD are commonly unstructured, using the connectivity of the mesh is not a feasible option and a novel approach to overimpose a *stencil*, with a fixed number of points, on the unstructured mesh is proposed. The network is designed to learn the relation between the solution at the stencil points at a given time step and the solution at the centre of the stencil and the next time step. Once trained, the network can be deployed in a degree adaptive process to predict the solution at the next time step before adapting the degree, thus preventing the loss of information in time and the need to repeat the time step to guarantee accuracy.

This chapter provides a brief overview of the ANN used in this work, including the activation functions considered and the optimisation algorithm used to train the network. Details about the normalisation of the data employed and the evaluation of the performance are also provided. Particular attention is given to the sampling of the parametric space using quasirandom Halton distributions and the treatment of the data before training the neural network. This treatment involves novel algorithms designed to reduce the amount of redundant information (e.g., corresponding to areas with free-stream flow), which not only helps decrease the amount of data used and the training time, but also removes potential bias in the trained network. Details are also given on the integration of the trained network within a CFD solver for real-time prediction and adaptivity.

Using machine learning techniques to predict flow behaviour, the aim is to demonstrate a significant improvement in degree adaptive simulation strategies, leading to more accurate and computationally efficient results. This predictive methodology represents a paradigm shift in the degree adaptive methods, utilising available data to speed up adaptive processes. Despite the current work focusing on degree adaptivity, it is worth mentioning that these ideas could also be exploited is mesh adaptivity strategies.

It is worth noting that the use of machine learning to aid in a mesh adaptive process has recently been considered in (Dzanic et al., 2024), where the authors use reinforced learning to predict local refinement policies. The idea is to predict the error in the solution and refine the path that the solution will follow in time and remesh in advance to avoid constant remeshing. In contrast to this work, the approach presented here focuses on degree adaptivity in a high-order context. Given the use of very high-order time integrators, it is crucial to ensure that the adaptivity is performed regularly to ensure accurate propagation of flow features. Furthermore, predicting the path of the solution to increase the degree in a large area would result in a large computational cost. Recent work on degree adaptation for high-order discontinuous Galerkin compressible simulations has also been considered in (Huergo et al., 2024), where the authors use reinforced learning to automate a degree adaptive process. This work considers a training process in 1D and its use in 2D and 3D problems to perform anisotropic degree adaptation. The main drawback is the use of tensor-product elements, given that nowadays it is still not possible to obtain unstructured meshes of hexahedral elements for complex geometries, and meshing complex models with structured hexahedral meshes requires a significant level of human intervention.

4.2 Artificial neural networks

ANNs are computational models inspired by biological neural networks of the human brain (Haykin, 2009). They consist of interconnected nodes or “neurons” organised in

layers that can learn complex non-linear relationships between inputs and outputs. The basic structure includes an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight that is adjusted during the training process.

Neural networks have shown remarkable success in various domains, including computer vision, natural language processing, and scientific computing (LeCun et al., 2015). Their ability to approximate complex functions makes them a suitable choice for predicting the time evolution of flow features.

4.2.1 ANN architecture

Figure 4.1 illustrates a schematic representation of a multi-layer perceptron, a prevalent type of ANN. This architecture comprises three primary components: the input layer, the hidden layers, and the output layer.

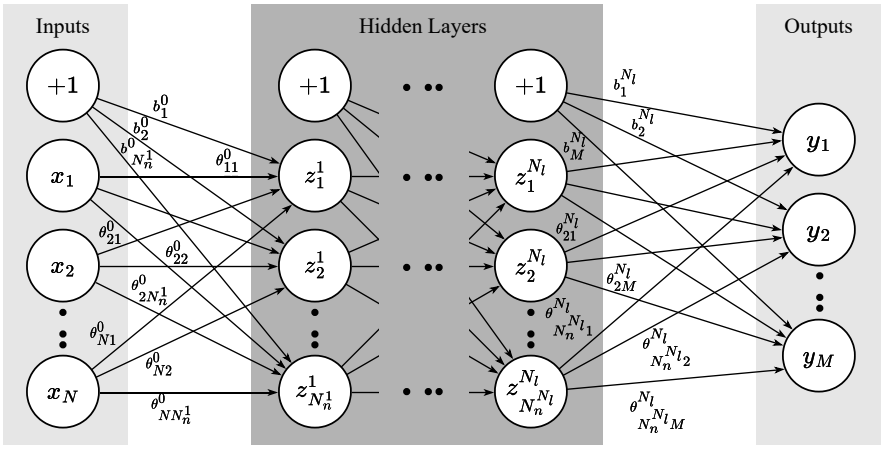


Figure 4.1: Schematic representation of a multi-layer perceptron neural network.

The input layer, depicted on the left side of Figure 4.1, consists of N input nodes $(x_1, x_2, \ldots, x_N)$ and a bias node (+1). These nodes represent the features or variables of the input data. The bias node, which always outputs a value of 1, allows the network to learn and represent patterns that do not necessarily pass through the origin.

The central portion of Figure 4.1 illustrates the hidden layers. An ANN may contain

multiple hidden layers, each comprising a different number of neurons. In this particular representation, we observe N_1 neurons $(z_1^1, z_2^1, \ldots, z_{N_1}^1)$ in the first hidden layer and N_l neurons $(z_1^{N_l}, z_2^{N_l}, \ldots, z_{N_l}^{N_l})$ in the final hidden layer. Each hidden layer also incorporates a bias node (+1).

The hidden layers are crucial for the ability of the ANN to model complex, non-linear relationships. Each successive layer allows the network to extract and combine features from the previous layer in increasingly abstract ways, thereby enabling the modelling of intricate patterns between inputs and outputs.

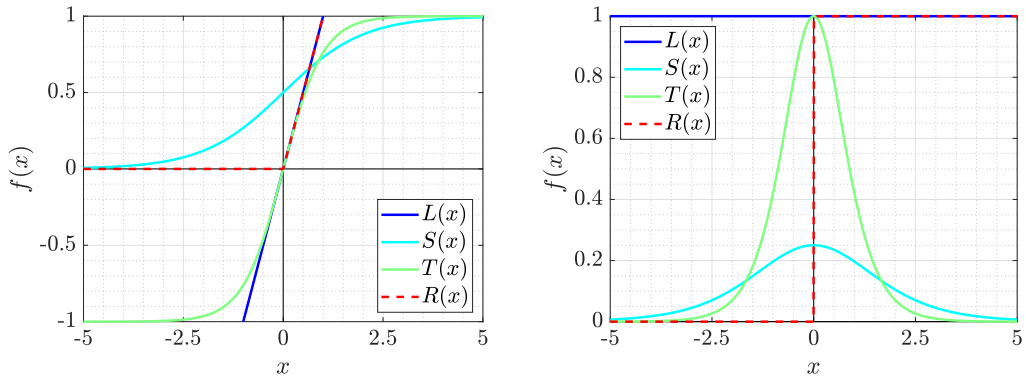
The right side of Figure 4.1 shows the output layer, comprising M output nodes $(y_1, y_2, \ldots, y_M)$. The number of output nodes is determined by the specific problem being addressed; for example, a binary classification task would typically have a single output node, while a multiclass classification problem would have multiple output nodes.

A key feature of ANNs is that the nodes in the adjacent layers are fully connected. A weight, denoted by θ in Figure 4.1, is associated to each connection. For example, θ_{ab}^c represents the weight of the connection between the a-th node of the c-th layer and the b-th connection of the subsequent layer.

4.2.2 Forward propagation

Information in an ANN flows from left to right, during the so-called forward propagation. The input values are propagated through the network by calculating the value associated to each neuron using a weighted sum of its inputs. Mathematically, for a neuron j in layer l, this computation can be expressed as

$$z_j^{l+1} = f^l \left(\sum_{i=1}^{N_n^l} \theta_{ji}^l z_i^l + b_j^l \right) \tag{4.1}$$



(a) Activation function (b) Derivative of activation function

Figure 4.2: The differences among four popular types of activation functions used in an ANN and their respective derivatives.

where f^l is the activation function of the l-th layer, θ^l_{ji} is the weight connecting neuron i in layer $l-1$ to neuron j in layer l, z^l_i is the output of neuron i in layer l, and b^l_j is the bias unit.

The activation functions are used to introduce the required non-linearities to ensure that the network can approximate complex relationships between inputs and outputs. Some of the most popular activation functions are:

Linear:	$L(x) = x$
Logistic Sigmoid:	$S(x) = \dfrac{1}{1+e^{-x}}$
Hyperbolic Tangent Sigmoid:	$T(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Rectified Linear Unit (ReLU):	$R(x) = \max(0, x)$

and are represented in Figure 4.2 together with their derivatives.

The choice of activation function can significantly affect network performance and training dynamics. It is common to use different activation functions in different layers of the network, tailored to the specific requirements of the task at hand.

Linear activation is typically used in the output layer for regression tasks. Sigmoid functions (logistic and hyperbolic tangent) are historically popular for their bounded

output and smooth gradients but can suffer from vanishing-gradient problems in deep networks. Finally, ReLU has become very popular in recent years, especially in deep learning, due to its computational efficiency and ability to mitigate the vanishing gradient problem. However, it is not differentiable at $x = 0$ and can lead to "dying ReLU" problems.

4.2.3 ANN training

The process of training an ANN involves adjusting the weights and biases to minimise the difference between the outputs predicted by the ANN and the true outputs, which are usually known for a few number of cases.

The so-called loss function, defined as

$$C(\boldsymbol{\theta}) = \frac{1}{\mathtt{n_{Tr}} M} \sum_{k=1}^{\mathtt{n_{Tr}}} \sum_{i=1}^{M} \left[y_i^k(x^k) - h_i^k(\boldsymbol{\theta}) \right]^2 \tag{4.2}$$

is introduced to measure the difference between true and predicted outputs, where $\mathtt{n_{Tr}}$ is the number of training examples, M is the number of outputs, y_i^k represents the true output for the i-th neuron and k-th training example, and $h_i^k(\boldsymbol{\theta})$ denotes the prediction of the network. The vector $\boldsymbol{\theta}$ encompasses all trainable parameters (weights and biases) of the network.

To minimise the loss function, i.e., to train the ANN, conjugate gradient-based approaches are commonly employed. Conjugate gradient methods are second-order optimisation techniques that aim to minimise a function by generating a sequence of search directions that are conjugate with respect to the Hessian matrix of the objective function.

Traditional conjugate gradient methods, such as the Fletcher-Reeves algorithm, require a line search procedure to determine the optimal step size in each iteration, which can be computationally expensive. The Scaled Conjugate Gradient (SCG) algorithm,

introduced by (Møller, 1993) and adopted in this work, eliminates the need for a user-specified line search by using a scaled step size. It combines the model-trust region approach from the Levenberg-Marquardt algorithm with the conjugate gradient approach, resulting in an efficient and robust optimisation method. The SCG algorithm has been shown to be particularly advantageous for training large-scale networks.

The SCG algorithm updates the weights at iteration $r + 1$ using the expression

$$\theta_{ij}^{l,r+1} = \theta_{ij}^{l,r} + \alpha^r p_{ij}^{l,r}, \tag{4.3}$$

where α^r is the step size determined by a model-trust region approach within the SCG method, and $p_{ij}^{l,r}$ is the conjugate direction, computed as

$$p_{ij}^{l,r} = -\nabla_\theta C(\boldsymbol{\theta}^r) + \beta^r p_{ij}^{l,r-1}, \tag{4.4}$$

with β^r given by

$$\beta^r = \frac{\nabla_\theta C(\boldsymbol{\theta}^r)^{\mathrm{T}} \left[\nabla_\theta C(\boldsymbol{\theta}^r) - \nabla_\theta C(\boldsymbol{\theta}^{r-1})\right]}{p_{ij}^{l,r-1} \left[\nabla_\theta C(\boldsymbol{\theta}^r) - \nabla_\theta C(\boldsymbol{\theta}^{r-1})\right]} \tag{4.5}$$

It can be observed that the SCG algorithm updates the weights using both first and second order derivatives of the cost function, resulting in faster convergence compared to first-order methods.

4.2.4 Performance Evaluation

To assess the performance of the trained network, a relative error metric, defined as

$$\text{Relative Error} = \frac{\max(|Y_{\text{test}} - Y_{\text{predicted}}|)}{\max(|Y_{\text{test}}|)} \tag{4.6}$$

is introduced.

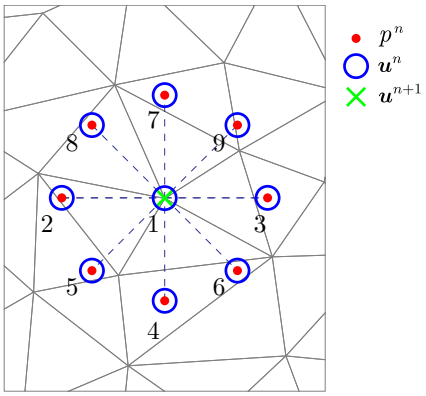


Figure 4.3: Schematic representation of the inputs and outputs for the proposed ANN architecture. Stencil with a central mesh node and eight neighbouring points. Input: consisting of p^n, and $\boldsymbol{u}^n$ and Output: consisting of $\boldsymbol{u}^{n+1}$ at the central mesh node.

This metric quantifies the maximum relative difference between the predicted and actual values for all test cases. It provides a non-dimensional measure of the worst case scenario and it will be used in the numerical examples to assess the accuracy of the trained ANNs.

4.3 Stencil-based ANN architecture

The proposed approach of using an ANN to predict the solution at time t^{n+1} from the solution at time t^n requires an architecture with a fixed number of inputs and outputs.

An obvious option is to consider all the nodes in the mesh and build a network that takes the solution at the current time step as input and predicts the solution at the future time step for all the nodes of the mesh. This architecture is expected to require a significant amount of data and training time due to the potentially large number of inputs and outputs. Furthermore, this strategy would not allow for training in one mesh and using the network in other meshes or even in the same mesh with different orders of approximation. Therefore, it is considered not a suitable option to couple with a degree adaptive approach.

The strategy considered consists of associating to each mesh node a set of surrounding points at a certain distance d, as shown in Figure 4.3. The plot shows a mesh node

(labelled '1') and eight surrounding points (labelled '2' to '8') at a certain distance d, the total number of points of the stencil is defined as $\mathtt{n}_{\mathtt{stencil}}$. The assumption here is that the solution at the central mesh node at t^{n+1} can be predicted from the solution at the surrounding points at t^n. More precisely, it is assumed that the velocity at the central mesh node can be predicted from the velocity at the same mesh node and at the surrounding points together with the pressure at the surrounding points only. This is schematically represented in Figure 4.3 and, in two dimensions, leads to an ANN architecture with $N = 26 + \mathtt{n}_{\mathtt{param}}$ inputs, where $\mathtt{n}_{\mathtt{param}}$ is the number of parameters characterising the flow and $M = 2$ outputs.

The main reason for not considering the pressure at the central mesh node as an input is because the incompressible Navier-Stokes equations only contain first order derivatives of the pressure. It is worth noting that it is only necessary to predict the velocity because the objective is to predict the degree of approximation, and the error indicator used in this work builds this error indicator only using the velocity field (and the postprocessed velocity).

This idea is inspired from numerical schemes such as the finite difference method where a stencil is typically built around each node of the Cartesian grid, so the set of points surrounding a node will be referred to as a the stencil of a mesh node. Different stencils have been considered, and numerical examples will present the performance of different strategies.

4.3.1 Stencil distance computation

The definition of the distance d used to locate the points on the stencil is based on physical considerations, mainly based on the hypothetical trajectory of a fluid particle over a single time step.

For each mesh node, let us consider the current position of a particle, $\boldsymbol{x}_i$, the velocity vector at that node, $\boldsymbol{u}_i$, and the time step, Δt . These parameters enable a simple

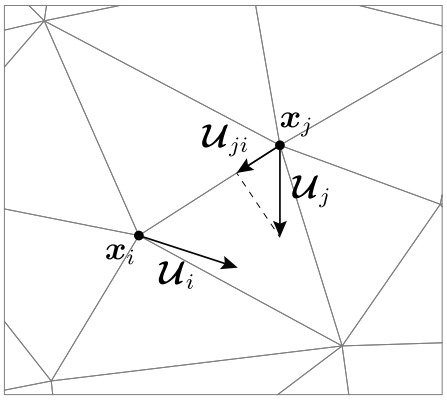


Figure 4.4: Schematic representation computation of the stencil distance for a node $\boldsymbol{x}_i$.

estimation of the displacement vector of a particle, defined as

$$\boldsymbol{\mathcal{U}}_i := \boldsymbol{u}_i \Delta t. \tag{4.7}$$

For each mesh node $\boldsymbol{x}_i$, a region of influence is defined using the magnitude of the displacement vector, $\|\boldsymbol{\mathcal{U}}_i\|_2$, representing the region from which information could potentially affect the solution at node $\boldsymbol{x}_i$ in the next time step.

In addition, the displacement vector for any mesh node $\boldsymbol{x}_j$ connected to node $\boldsymbol{x}_i$ is projected over the edge connecting nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, namely

$$\boldsymbol{\mathcal{U}}_{ji} := \frac{\boldsymbol{\mathcal{U}}_j \cdot \boldsymbol{x}_{ji}}{\|\boldsymbol{x}_{ji}\|_2} \boldsymbol{x}_{ji}, \tag{4.8}$$

where $\boldsymbol{x}_{ji} := \boldsymbol{x}_j - \boldsymbol{x}_i$, as illustrated in Figure 4.4.

The distance associated to node $\boldsymbol{x}_i$ is then defined as

$$d_i := \min \left\{ \|\boldsymbol{\mathcal{U}}_i\|_2, \min_{j \in \Lambda_i} \{ \|\boldsymbol{\mathcal{U}}_{ji}\|_2 \} \right\} \tag{4.9}$$

where Λ_i denotes the set of nodes connected to node $\boldsymbol{x}_i$.

This approach ensures the application of the most restrictive condition, particularly in regions exhibiting high velocity gradients. This distance computation methodology is closely aligned with semi-Lagrangian numerical schemes. In these schemes, the

solution at a point is computed by tracing the characteristics backward in time. The distances computed in our algorithm effectively estimate the length of these characteristic lines, adapting the numerical stencil to the local flow behaviour.

The distance d_i defined in Equation (4.9) could be added to the list of inputs of the ANN to be trained. However, to avoid adding extra parameters and minimise the amount of data required and training time, a unique value defined as the minimum for all mesh nodes, that is

$$d := \min_{i=1,\dots,\mathtt{n_{nodes}}} \{d_i\}, \tag{4.10}$$

is considered in this work.

4.3.2 Data acquisition

The generation of suitable data to train an ANN requires sampling the space of the parameters. Typical methods to perform this task are the Latin hypercube sampling (LHS) (Shields and Zhang, 2016) and the Halton sequences (Halton, 1960).

The Halton sequence is a deterministic sequence of points that provides a uniform distribution over the unit hypercube $[0, 1]^N$, where N is the dimension of the space. Halton sequences are considered in this work because they offer better uniformity and coverage of the sample space compared to pseudo-random generators, especially in lower dimensions (Kocis and Whiten, 1997), and contrary to random methods such as LHS the Halton sequencing enables repeatability. The last aspect is considered important when performing studies to analyse the influence of the number of training cases on the accuracy of the ANN predictions.

For a given base b, the one-dimensional Halton sequence is generated as

$$H_b(m) = \sum_{i=0}^{\infty} a_i(m) b^{-(i+1)} \tag{4.11}$$

where m is the index of the sequence (starting from 0), and $a_i(m)$ are the digits of m

when expressed in base b, but in reverse order.

To generate a N-dimensional Halton sequence, we use N different prime numbers, typically the first N prime numbers, as bases for each dimension.

The data acquisition process for the proposed ANN comprises two main steps. The first one is the the execution of a series simulations for some varying parameters, typically corresponding to different flow conditions. The pressure (p) and velocity ($\boldsymbol{u}$) fields are stored in an four dimensional array $\boldsymbol{\mathcal{X}}$ of dimension $\mathtt{n}_{\mathtt{nodes}} \times \mathtt{n}_{\mathtt{steps}} \times (\mathtt{n}_{\mathtt{sd}} + 1) \times \mathtt{n}_{\mathtt{Tr}}$, where $\mathtt{n}_{\mathtt{nodes}}$ is the number of mesh nodes, $\mathtt{n}_{\mathtt{steps}}$ is the number of time steps and $\mathtt{n}_{\mathtt{Tr}}$ denotes the number of training cases.

Each component of the array $\mathcal{X}_{a,b,c,d}$ contains the c-th component of the velocity for $c = 1, \ldots, \mathtt{n}_{\mathtt{sd}}$ and the pressure for $c = \mathtt{n}_{\mathtt{sd}} + 1$ at the a-th mesh node and the b-th time step and for the set of flow parameters corresponding to the d-th training case.

The second step consists of transforming the data stored in $\boldsymbol{\mathcal{X}}$ into two two-dimensional arrays $\mathbf{X}$ and $\mathbf{Y}$ to be used as input and output of the ANN. The input array $\mathbf{X}$ is of dimension $\mathtt{n}_{\mathtt{nodes}}\mathtt{n}_{\mathtt{steps}}\mathtt{n}_{\mathtt{Tr}} \times N$, with each row storing the following information

$$\begin{aligned}\mathbf{X}_r = \big\{ & p^n_{i,2}(\boldsymbol{\mu}),\ p^n_{i,3}(\boldsymbol{\mu}),\ \ldots,\ p^n_{i,9}(\boldsymbol{\mu}),\ u^n_i(\boldsymbol{\mu}),\ u^n_{i,2}(\boldsymbol{\mu}),\ u^n_{i,3}(\boldsymbol{\mu}),\ \ldots,\ u^n_{i,9}(\boldsymbol{\mu}), \ldots \\ & v^n_i(\boldsymbol{\mu}),\ v^n_{i,2}(\boldsymbol{\mu}),\ v^n_{i,3}(\boldsymbol{\mu}),\ \ldots,\ v^n_{i,9}(\boldsymbol{\mu}),\ \boldsymbol{\mu} \big\},\end{aligned} \tag{4.12}$$

for a given mesh node $\boldsymbol{x}_i$ and instant t^n, where $\boldsymbol{\mu} := \{\mu_1,\ \ldots,\ \mu_{\mathtt{n}_{\mathtt{param}}}\}$ denotes the parameters encoding the flow conditions, $\mathtt{n}_{\mathtt{param}}$ is the number of parameters and $\square^n_{i,a}$ denotes the interpolation of $\square$ at the a-th point of stencil centred at $\boldsymbol{x}_i$ at time t^n. The row number r is given by

$$r = (i_{\mathtt{Tr}} - 1)\mathtt{n}_{\mathtt{nodes}}\mathtt{n}_{\mathtt{steps}} + (n-1)\mathtt{n}_{\mathtt{nodes}} + i \tag{4.13}$$

where $i_{\mathtt{Tr}}$ denotes the i-th training case.

Similarly, the output array $\mathbf{Y}$ is of dimension $\mathtt{n}_{\mathtt{nodes}}\mathtt{n}_{\mathtt{steps}} \times M$, with each row storing the following information

$$\mathbf{Y}_r = \left\{u_i^{n+1},\ v_i^{n+1}\right\}. \tag{4.14}$$

The methodology is illustrated in Algorithm 3. For each node $\boldsymbol{x}_i$ of the computational mesh and for each instant t^n the solution is interpolated at the points of the stencil corresponding to $\boldsymbol{x}_i$. The interpolation requires finding the *parent* element of the mesh that contains the point of the stencil, computing the local coordinates in the reference element for this point at the identified parent element and interpolating the solution using the corresponding high-order shape functions. The interpolated values, together with the associated parameters, form the row of the input array as described in Equation (4.12). For the output array, the values are simply the components of the velocity at the next instant t^{n+1} as detailed in Equation (4.14).

For nodes on the boundary or close to the boundary, it is possible to have stencil points outside of the computational domain. These nodes are discarded and are not used to build a stencil. This implies that the proposed approach assumes that the mesh and the degree of approximation near the boundary are appropriate. This assumption is well-founded, as practitioners have developed robust mesh generation expertise for steady-state simulations over several decades. Experience has demonstrated that targeted refinement near boundaries effectively captures critical flow features like boundary layers and separation points. This is especially relevant for external aerodynamics and internal flows where the most significant gradients develop near solid boundaries. While adaptive refinement remains valuable, starting with appropriately refined boundary meshes based on established engineering practice has proven reliable. The objective of the proposed approach is to perform degree adaptive to capture flow features travelling long distances within the domain.

The interpolation process when a point of the stencil lies within an element that is affine to the reference element is trivial, as inverting the isoparametric mapping is trivial. However, when a point of the stencil lies within an element that is not affine

to the reference element (e.g. a high-order curved triangle in two dimensions) the inversion of the isoparametric mapping requires the solution of a small non-linear problem to find the local coordinates of the point as explained in Section 2.6

4.3.3 Data preparation

Data normalisation is a crucial preprocessing step in training ANNs. It helps to ensure that all input and outputs are on a similar scale, which can significantly improve the performance and convergence of the training process. One common technique for normalisation is the Min-Max normalisation, also known as feature scaling, which involves scaling all values to a fixed range, typically between 0 and 1. This transformation is particularly useful when the features have different scales or units.

For a given quantity z, the min-max normalisation is defined as:

$$z_{\text{normalised}} = \frac{z - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \tag{4.15}$$

where z_{min} and z_{max} are the minimum and maximum values of the quantity z in the dataset, respectively.

When applying the Min-Max Normalisation to the inputs of an ANN, it is crucial to ensure that the minimum and maximum values for each input are computed using only the training data. Then, the normalisation is applied to training, validation and test sets using Equation (4.15). This procedure ensures that the training process does not see any information from validation or test sets.

Another crucial aspect of the data preparation process, which is specific to the application of interest, involves discarding regions that do not exhibit any relevant transient phenomena, that is, regions where the flow is almost uniform. This process not only reduces the amount of data stored and used for training but also ensures that the training is not biased towards those cases corresponding to uniform flow inputs and outputs. To

this end, a filter tolerance, denoted ε_∇ is introduced and nodes where the gradient of the pressure and velocity are both below the tolerance are discarded from the training dataset.

Formally, we discard a point $\boldsymbol{x}_i$ at time t^n if

$$\max\left\{\left\|\boldsymbol{\nabla} p(\boldsymbol{x}_i, t^n)\right\|_2, \left\|\boldsymbol{\nabla} u(\boldsymbol{x}_i, t^n)\right\|_2, \left\|\boldsymbol{\nabla} v(\boldsymbol{x}_i, t^n)\right\|_2\right\} \leq \varepsilon_\nabla, \tag{4.16}$$

where $\|\cdot\|_2$ denote the Euclidean norm.

The data acquisition and filtering process is detailed in Algorithm 3. The first loop

Algorithm 3 Data acquisition and filtering

Input: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathtt{n_{nodes}} \times \mathtt{n_{steps}} \times (\mathtt{n_{sd}}+1) \times \mathtt{n_{Tr}}}$
Output: $\mathbf{X} \in \mathbb{R}^{\mathtt{n_{cases}} \times N}$, $\mathbf{Y} \in \mathbb{R}^{\mathtt{n_{cases}} \times M}$
$\mathtt{n_{cases}} = 0$ {Initialise case counter}
for $i_{\mathtt{sim}} \leftarrow 1$ **to** $\mathtt{n_{sim}}$ **do**
  for $n \leftarrow 1$ **to** $\mathtt{n_{steps}} - 1$ **do**
    for $i \leftarrow 1$ **to** $\mathtt{n_{nodes}}$ **do**
      if $\max\left\{\left\|\boldsymbol{\nabla} p(\boldsymbol{x}_i, t^n)\right\|_2, \left\|\boldsymbol{\nabla} u(\boldsymbol{x}_i, t^n)\right\|_2, \left\|\boldsymbol{\nabla} v(\boldsymbol{x}_i, t^n)\right\|_2\right\} > \varepsilon_\nabla$ **then**
        $\mathcal{S} \leftarrow \{\boldsymbol{x}_{i,s}\}_{s=2}^{\mathtt{n_{stencil}}}$ {Coordinates for the stencil points centered at $\boldsymbol{x}_i$}
        if $\mathcal{S} \subset \Omega$ **then**
          for $s \leftarrow 2$ **to** $\mathtt{n_{stencil}}$ **do**
            Interpolate solution at stencil point $\boldsymbol{x}_{i,s}$
          end for
          $\mathtt{n_{cases}} \leftarrow \mathtt{n_{cases}} + 1$
          Set r according to Equation 4.13
          Set $\mathbf{X}_r$ according to Equation 4.12
          Set $\mathbf{Y}_r$ according to Equation 4.14
          $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{X}_r$
          $\mathbf{Y} \leftarrow \mathbf{Y} \cup \mathbf{Y}_r$
        end if
      end if
    end for
  end for
end for
return $\mathbf{X} \in \mathbb{R}^{\mathtt{n_{cases}} \times N}$, $\mathbf{Y} \in \mathbb{R}^{\mathtt{n_{cases}} \times M}$

considers all the parametric cases available. For each parametric case a loop on time steps and mesh nodes is performed. For each mesh node the gradients of the gradients of the pressure and velocity are checked to ensure that only relevant cases are considered. For relevant cases the stencil points are computed and, if the whole stencil lies within

the computational domain, the solution is interpolated at each stencil point. With this information, the matrices of inputs and outputs are populated.

Once all data have been collected, it is essential to ensure that there are no duplicate input cases. The presence of repeated data can introduce significant redundancy into the dataset, potentially skewing the training process and leading to a biased neural network model that overfits to these repetitive patterns. This step is crucial to maintain the diversity and representativeness of the dataset.

An algorithm has been developed and implemented to reduce the data set by eliminating redundant cases. A case is considered redundant if, at a given mesh point, the variables at each node of the stencil are sufficiently close to the corresponding values at another mesh point.

In the current implementation, the range of each input variable is calculated at each node within the stencil and subsequently divided into a predetermined number of intervals. More precisely, for each input the minimum and maximum values found for all cases are defined as

$$\alpha_i^{\mathbf{X}} := \min_{a=1,\dots,\mathtt{n}_{\mathtt{cases}}} \{\mathbf{X}_{a,i}\}, \qquad \beta_i^{\mathbf{X}} := \max_{a=1,\dots,\mathtt{n}_{\mathtt{cases}}} \{\mathbf{X}_{a,i}\}. \tag{4.17}$$

Each interval $\left[\alpha_i^{\mathbf{X}}, \beta_i^{\mathbf{X}}\right]$ is divided into $\mathtt{n}_{\mathtt{buckets}}$ equally-spaced intervals that are referred to as *buckets* and denoted by

$$\mathbf{B}_i^{\mathbf{X}} = \{b_{i,1}^{\mathbf{X}}, \dots, b_{i,\mathtt{n}_{\mathtt{buckets}}}^{\mathbf{X}}\}. \tag{4.18}$$

In a similar fashion, for each output the minimum and maximum values found for all cases are defined as

$$\alpha_i^{\mathbf{Y}} := \min_{a=1,\dots,\mathtt{n}_{\mathtt{cases}}} \{\mathbf{Y}_{a,i}\}, \qquad \beta_i^{\mathbf{Y}} := \max_{a=1,\dots,\mathtt{n}_{\mathtt{cases}}} \{\mathbf{Y}_{a,i}\}. \tag{4.19}$$

Each interval $\left[\alpha_i^{\mathbf{Y}}, \beta_i^{\mathbf{Y}}\right]$ is divided into the same number of *buckets*, denoted by

$$\mathbf{B}_i^{\mathbf{Y}} = \{b_{i,1}^{\mathbf{Y}}, \ldots, b_{i,\mathtt{n}_{\mathtt{buckets}}}^{\mathbf{Y}}\}. \tag{4.20}$$

Each variable at a node is then categorised into one of these buckets. A specific case is identified by the unique combination of buckets occupied by the variables at the nodes of the stencil. Consequently, a case is considered redundant if the same combination of buckets has already been encountered in the data set. This method ensures the removal of repeated cases, thus improving the efficiency of the data set and improving the overall quality of the ANN training process. This procedure is detailed in Algorithm 4.

4.3.4 Degree adaptive HDG strategy using prediction

Algorithm 5 presents a degree-adaptive method incorporating an Artificial Neural Network (ANN). The primary enhancements to the standard adaptive strategy, highlighted in green, involve the use of predictive techniques. These modifications enable prediction of the velocity and superconvergent velocity for the subsequent time step, thereby enabling the computation of an error indicator for the predicted solution. The polynomial degree is subsequently updated by considering the maximum of both the computed and predicted error indicators. This approach potentially enhances the efficiency and accuracy of the method for unsteady problems by adapting the polynomial degree based on both current and anticipated future solution behaviour.

4.4 Verification example

As discussed in Chapter 3, adaptivity constitutes a significant challenge in the context of transient flows. The numerical example in Section 3.5.2 demonstrated that when using high order time integrators and large time steps for maximum efficiency, an adaptive process fails to capture accurately the propagation of vortices, and this is reflected in

Algorithm 4 Data reduction

$\mathbf{X} \in \mathbb{R}^{\mathtt{n}_{\mathtt{cases}} \times N}$, $\mathbf{Y} \in \mathbb{R}^{\mathtt{n}_{\mathtt{cases}} \times M}$
Output: $\mathbf{X}^{\mathtt{red}} \in \mathbb{R}^{\mathtt{n}_{\mathtt{red}} \times N}$, $\mathbf{Y}^{\mathtt{red}} \in \mathbb{R}^{\mathtt{n}_{\mathtt{red}} \times M}$
$\mathtt{n}_{\mathtt{red}} = 0$ {Initialise reduced case counter}
Define buckets for inputs and outputs according to Equations (4.18) and (4.20)
Initialise alternating digital tree (ADT): $\mathcal{T} \leftarrow \emptyset$
for $\mathtt{i}_{\mathtt{c}} \leftarrow 1$ **to** $\mathtt{n}_{\mathtt{cases}}$ **do**
  for $\mathtt{i}_{\mathtt{out}} \leftarrow 1$ **to** M **do**
    Find j such that $b^{\mathbf{Y}}_{i,j} \leq \mathrm{Y}_{\mathtt{i}_{\mathtt{c}},\mathtt{i}_{\mathtt{out}}} \leq b^{\mathbf{Y}}_{i,j+1}$
  end for
  if $\mathtt{i}_{\mathtt{c}} > 1$ **then**
    $\mathbf{Z} \leftarrow \{z \in \mathcal{T} : b^{\mathbf{Y}}_{i,j} \leq z \leq b^{\mathbf{Y}}_{i,j+1}, \mathtt{i}_{\mathtt{out}} = 1, \ldots, M\}$
    {The ADT returns the output cases in which all the outputs lie on the same bucket as case $\mathtt{i}_{\mathtt{c}}$}
    $r \leftarrow 0$
    for $z \in \mathbf{Z}$ **do**
      for $\mathtt{i}_{\mathtt{in}} \leftarrow 1$ **to** N **do**
        Find l such that $b^{\mathbf{X}}_{i,l} \leq \mathrm{X}_{\mathtt{i}_{\mathtt{c}},\mathtt{i}_{\mathtt{in}}} \leq b^{\mathbf{X}}_{i,l+1}$
        Find l_z such that $b^{\mathbf{X}}_{i,l_z} \leq \mathrm{X}_{z,\mathtt{i}_{\mathtt{in}}} \leq b^{\mathbf{X}}_{i,l_z+1}$
        if $l \neq l_z$ **then**
          $r \leftarrow r + 1$
          break
        end if
      end for
    end for
    if $r = 0$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{i}_{\mathtt{c}}\}$
      $\mathbf{X}^{\mathtt{red}} \leftarrow \mathbf{X}^{\mathtt{red}} \cup \mathbf{X}_{\mathtt{i}_{\mathtt{c}},\star}$
      $\mathbf{Y}^{\mathtt{red}} \leftarrow \mathbf{Y}^{\mathtt{red}} \cup \mathbf{Y}_{\mathtt{i}_{\mathtt{c}},\star}$
      $\mathtt{n}_{\mathtt{red}} = \mathtt{n}_{\mathtt{red}} + 1$
    end if
  else
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{i}_{\mathtt{c}}\}$
    $\mathbf{X}^{\mathtt{red}} \leftarrow \mathbf{X}^{\mathtt{red}} \cup \mathbf{X}_{\mathtt{i}_{\mathtt{c}},\star}$
    $\mathbf{Y}^{\mathtt{red}} \leftarrow \mathbf{Y}^{\mathtt{red}} \cup \mathbf{Y}_{\mathtt{i}_{\mathtt{c}},\star}$
    $\mathtt{n}_{\mathtt{red}} = \mathtt{n}_{\mathtt{red}} + 1$
  end if
end for
return $\mathbf{X}^{\mathtt{red}} \in \mathbb{R}^{\mathtt{n}_{\mathtt{red}} \times N}$, $\mathbf{Y}^{\mathtt{red}} \in \mathbb{R}^{\mathtt{n}_{\mathtt{red}} \times M}$

Algorithm 5 Degree adaptive HDG Method for unsteady Navier-Stokes Equations

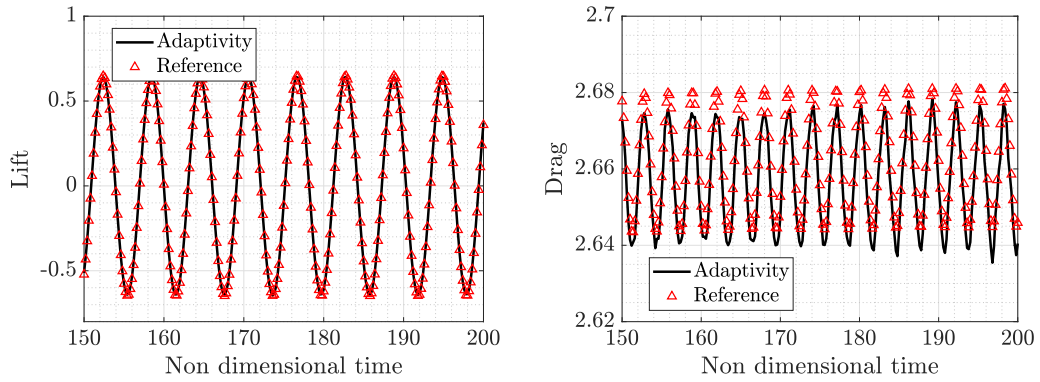
Initialise polynomial degree map $\{k_e\}_{e=1,\ldots,\mathtt{n_{el}}}$
Set base b for logarithm, tolerance ε and number of iterations $\mathtt{n_{adaptivity}}$.
for $i_\mathtt{s} \leftarrow 1$ **to** $\mathtt{n_{steps}}$ **do**
  for $i_\mathtt{a} \leftarrow 1$ **to** $\mathtt{n_{adaptivity}}$ **do**
    for $i_\mathtt{NR} \leftarrow 1$ **to** $\mathtt{n_{NR}}$ **do**
      Solve global problem of Equation (2.51)
      Solve local problem of Equation (2.53)
    end for
    for $i_\mathtt{el} \leftarrow 1$ **to** $\mathtt{n_{el}}$ **do**
      Compute super-convergent velocity using Equation (3.2)
      Compute error indicator using Equation (3.11)
      Predict velocity and super-convergent velocity at time $i_\mathtt{s} + 1$ using ANN
      Compute error indicator for predicted solution using Equation (3.11)
      Update the degree using Equation (3.17), taking maximum of computed and predicted
      if $\Delta k_e < 0$ **then**
        Compute conservative projection using Equation (3.25)
      end if
    end for
  end for
end for

a substantial loss of accuracy in aerodynamic quantities of interest such as the drag. This example also showed that, to ensure accurate computation, the adaptive process required the repetition of each time step. This was shown to be enough to ensure that the degree of approximation was adapted in regions where flow features were not present at the time t^n but will be present at the next instant t^{n+1}.

4.4.1 Problem description

As a verification of the proposed use of ANN to aid in a degree adaptive process, this section considers the same example, i.e., the flow at $Re = 100$ around two cylinders in tandem, where the ANN is used to predict the solution at time t^{n+1} from the solution at time t^n. The mesh considered is the same as the previous example, shown in Figure 3.8 and the ESDIRK46 time marching is used with a time step $\Delta t = 0.2$

No flow parameters are considered here as the objective is simply to illustrate the potential benefits of having an ANN trained to accurately predict the solution at the



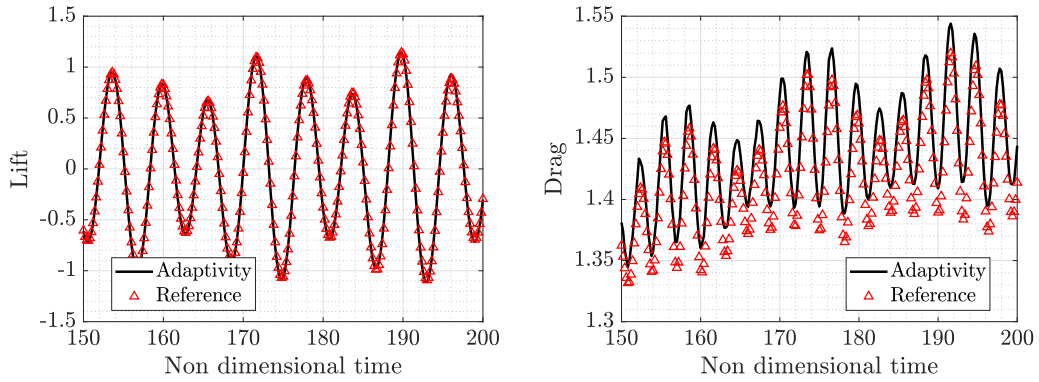
(a) Lift (b) Drag

Figure 4.5: Flow around two circular cylinders: lift and drag over the first cylinder using degree adaptivity compared to the reference solution.

next time step. Therefore, the data acquisition is performed using the simulation corresponding to the reference solution, which involves using a constant degree of approximation $k = 6$ in all elements. Despite this is purely an academic exercise to verify the correct coupling of the trained ANN within the degree adaptive loop, it is worth mentioning that this does not imply that the ANN is deployed to predict only cases that were seen during training. This is because the ANN is trained using data from a simulation with $k = 6$ in all elements, whereas the degree adaptive simulation involves degree adaptivity. Therefore, different values of the inputs corresponding to the flow field at a point and its corresponding stencil, when compared to the training cases, will be used to perform the predictions.

To test the proposed methodology, a standard degree adaptive process is first utilised with a target error of $\varepsilon = 0.5 \times 10^{-4}$. By standard, we refer to the application of the adaptivity without repeating the time step or involving the ANN prediction, but employing the conservative projection proposed in Section 3.4.2. The corresponding lift and drag for the first cylinder are illustrated in Figure 4.5, while those for the second cylinder are depicted in Figure 4.6.

The results clearly show an important loss of accuracy in the drag force for the first cylinder and, even more pronounced, for the second cylinder. As discussed in Section 3.5.2 the accurate results obtained for the lift are only due to the cancellation of errors



(a) Lift (b) Drag

Figure 4.6: Flow around two circular cylinders: lift and drag over the second cylinder using degree adaptivity and the proposed correction compared to the reference solution.

that results from the lift being centred at zero.

4.4.2 Data acquisition

To illustrate the potential of the proposed use of an ANN, the data acquisition is performed from the $\mathtt{n}_{\mathtt{steps}} = 250$ time steps of the reference solution, corresponding to $t \in (150, 200)$, following the procedure described in Section 4.3.2 and Algorithm 3. The stencil distance is computed following the procedure described in Section 4.3.1, leading to a value of $d = 0.0375$.

Given that flow parameters are not considered in this example, the ANN architecture comprises $N = 26$ inputs and $M = 2$ outputs, as detailed in Section 4.3.

The data acquisition process leads to a total of $\mathtt{n}_{\mathtt{cases}} = 20,532,381$ cases, mainly due to the use of a high-order reference solution with $k = 6$ in all elements. To reduce this large dataset and eliminate redundant information that can lead to a biased ANN, the data preparation procedure described in Section 4.3.3 is applied with $\mathtt{n}_{\mathtt{buckets}} = 10, 20$ and 40 buckets.

Figures 4.7, 4.8, and 4.9 show the data collected from the reference solution. In the horizontal axis the bucket number is displayed, whereas the vertical axis shows the cumulative number of cases. Each colour represents one of the inputs of the ANN and

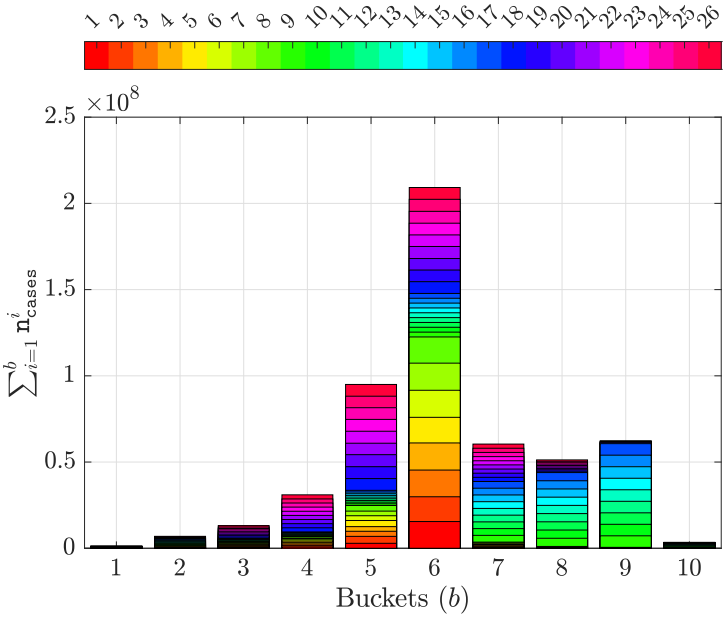


Figure 4.7: Flow around two circular cylinders: Histogram illustrating the distribution of the cases in 10 buckets for the data collected from a simulation and before any preparation.

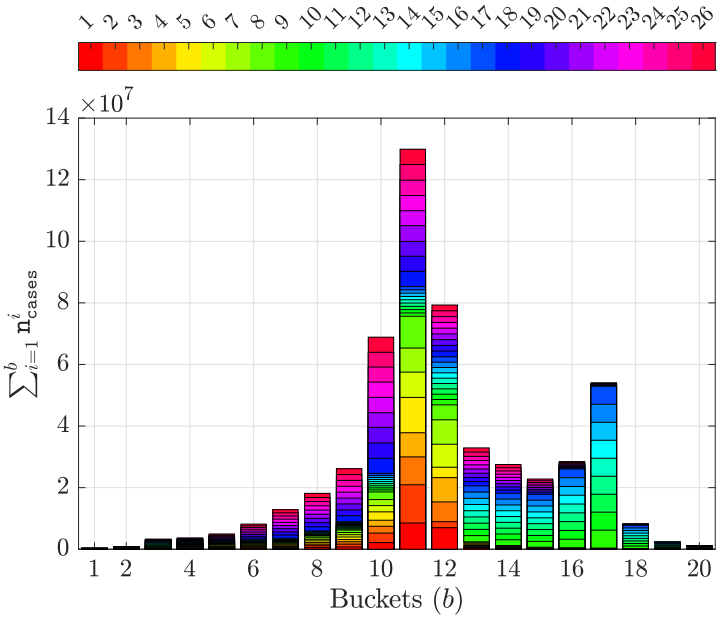


Figure 4.8: Flow around two circular cylinders: Histogram illustrating the distribution of the cases in 20 buckets for the data collected from a simulation and before any preparation.

the definition of each input is as described in Equation (4.12). The distribution among the buckets is clearly uneven, and the central buckets contain the highest number of cases, approaching 10^8 cases. In contrast, the buckets at the lower and higher ends of

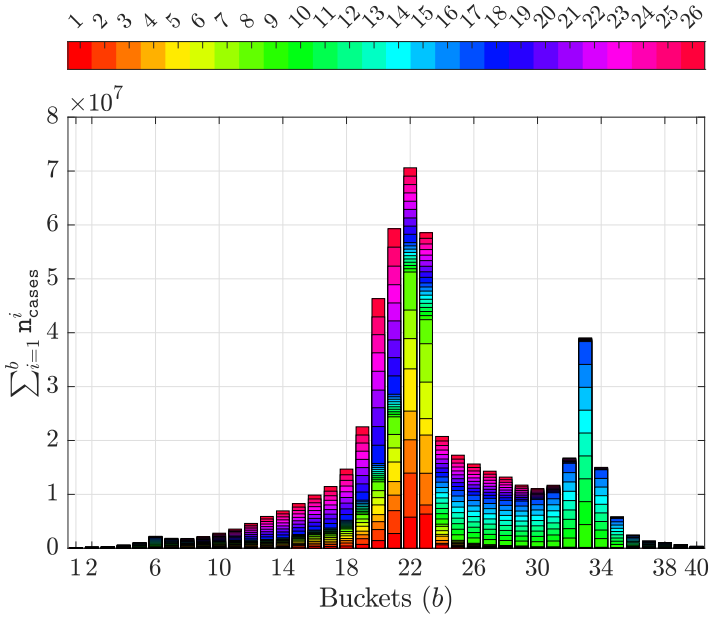


Figure 4.9: Flow around two circular cylinders: Histogram illustrating the distribution of the cases in 40 buckets for the data collected from a simulation and before any preparation.

the spectrum show significantly fewer cases, often below 10^6.

4.4.3 Data preparation

Given the large number of cases obtained from a single simulation, training directly with the data gathered is not desirable. On one hand, the training process is expected to take significant time and, on the other hand, the trained ANN is expected to be biased given the data distribution shown in Figure 4.8. To mitigate this problem, the data reduction process described in Section 4.3.3 is considered next.

Figures 4.10, 4.11, and 4.12 depict the distribution of the data after the data reduction process. The distribution among the buckets is significantly more uniform, with most of the buckets containing thousands of cases. It is also important to note the even distribution of colours (inputs) in each bucket, illustrating the performance of the data reduction algorithm proposed in Section 4.3.3. For this example, the total number of cases after reduction is: for 10 buckets is 535, for 20 buckets is 4,535, and for 40 buckets is 32646. This significant reduction in the dataset is expected to have a major impact

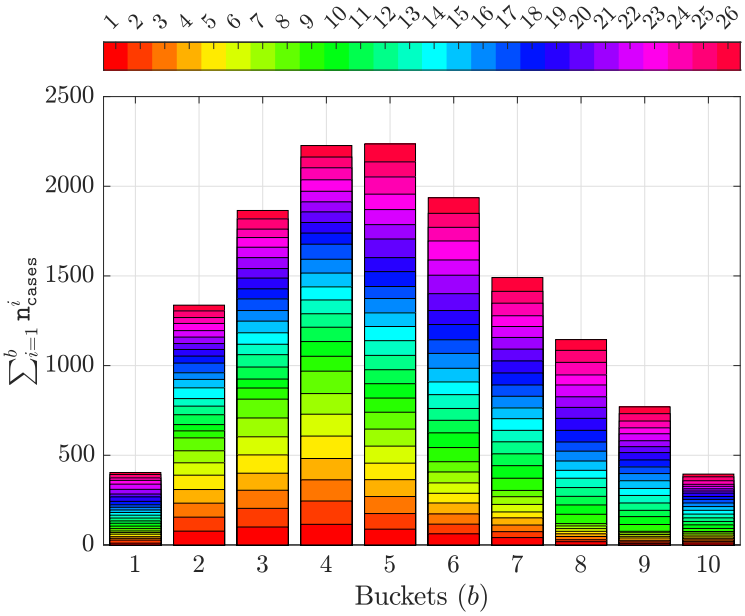


Figure 4.10: Flow around two circular cylinders: Histogram illustrating the distribution of cases in 10 buckets for the data collected from a simulation after the application of the data reduction algorithm.

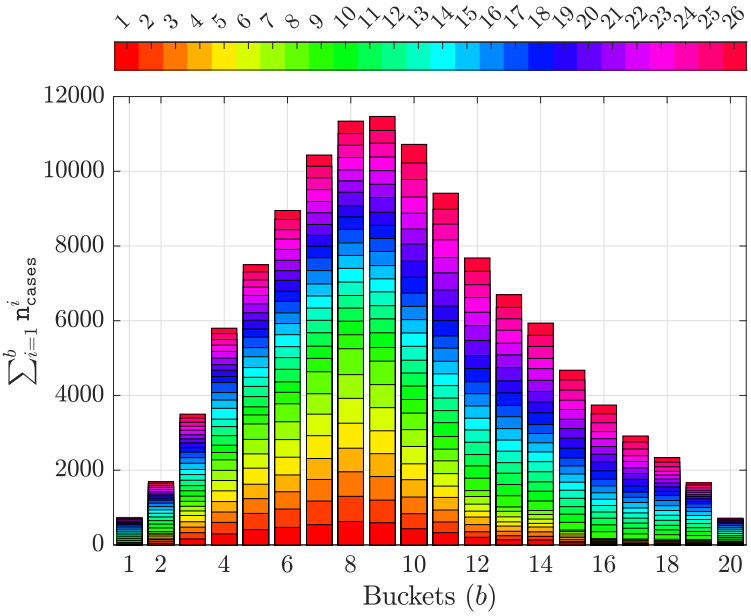


Figure 4.11: Flow around two circular cylinders: Histogram illustrating the distribution of cases in 20 buckets for the data collected from a simulation after the application of the data reduction algorithm.

on the training time of the ANN as well as the reduction of bias in the trained ANN.

Subsequent analyses will be performed using the data produced using 20 buckets.

To further analyse the performance of the reduction algorithm, Figure 4.13 compares

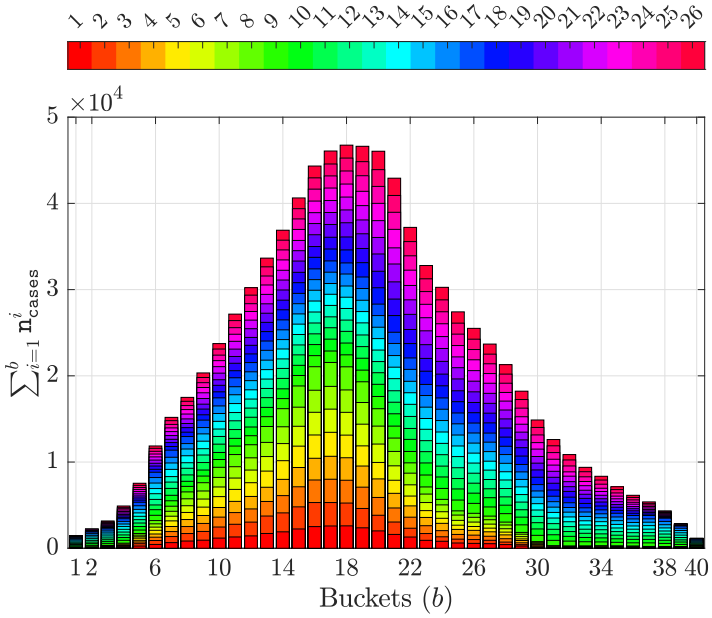
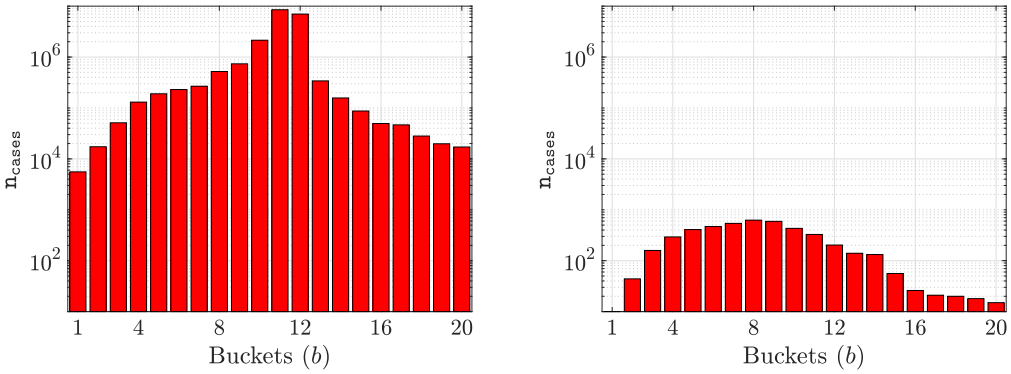


Figure 4.12: Flow around two circular cylinders: Histogram illustrating the distribution of cases in 40 buckets for the data collected from a simulation after the application of the data reduction algorithm.

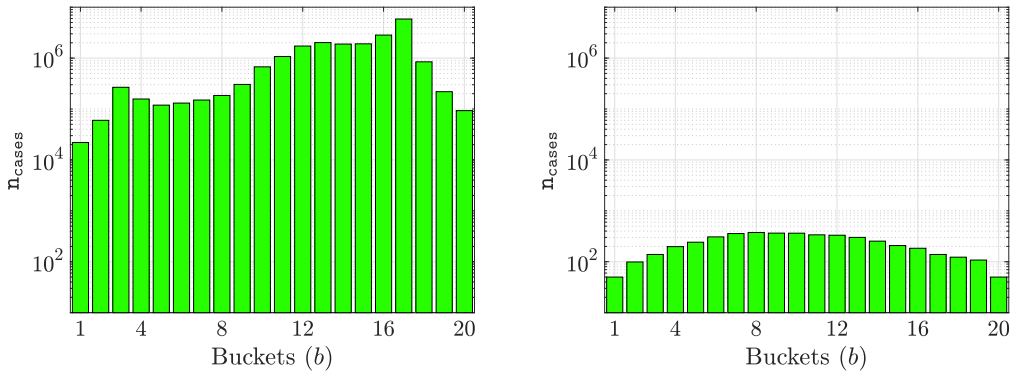


(a) Before data reduction

(b) After data reduction

Figure 4.13: Flow around two circular cylinders: Logarithmic histogram illustrating the distribution of cases in 20 buckets for the pressure field at the central point of the stencil before and after data reduction.

the distribution of the data corresponding to the pressure field at the central point of the stencil before and after the reduction process is applied. The results, presented in logarithmic scale clearly show that the raw data contains a significant number of cases on buckets 10 to 12, with more than a million cases on each of these buckets. After reduction, the distribution of the data is more uniform, with all buckets containing between 10 and 1,000 cases.



(a) Before data reduction

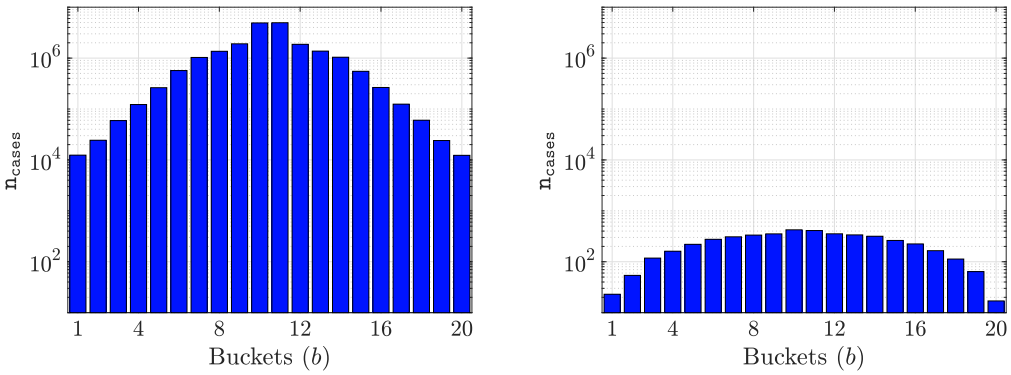
(b) After data reduction

Figure 4.14: Flow around two circular cylinders: Logarithmic histogram illustrating the distribution of cases in 20 buckets for the horizontal velocity at the central point of the stencil before and after data reduction.

Similarly, Figure 4.14 compares the distribution of the data corresponding to the horizontal velocity at the central point of the stencil before and after the reduction process is applied. The results show that the raw data contains a significant number of cases on buckets 12 to 17, with more than a million cases on each of these buckets. After reduction, the distribution of the data is more uniform, with all buckets containing between 10 and 1,000 cases, which is of the same order of cases when compared to the pressure field.

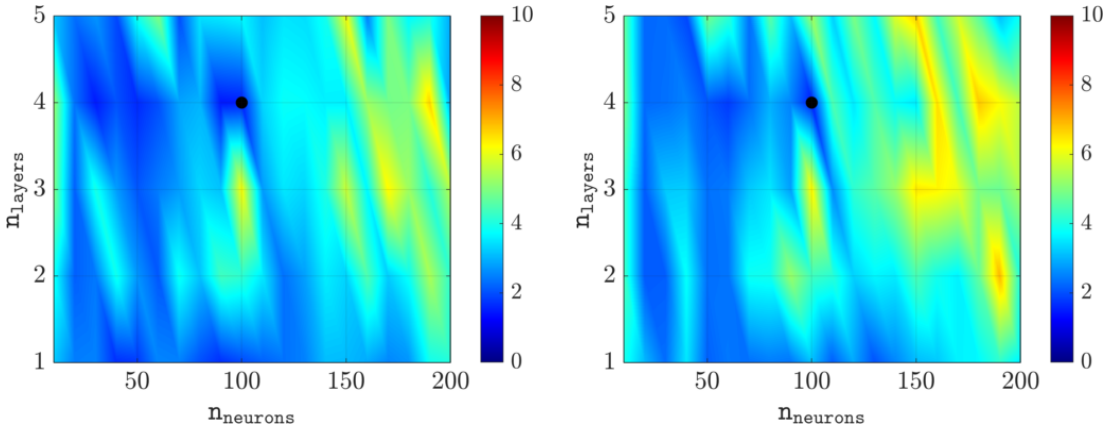
Finally, Figure 4.15 compares the distribution of the data corresponding to the vertical velocity at the central point of the stencil before and after the reduction process is applied.

It is worth noting that the raw data is always skewed but in different buckets, due to the different nature of the quantity being considered and the proposed algorithm produces a reduced dataset where all the variables show a significant reduction and a more uniform representation of the dataset.



(a) Before data reduction (b) After data reduction

Figure 4.15: Flow around two circular cylinders: Logarithmic histogram illustrating the distribution of cases in 20 buckets for the vertical velocity at the central point of the stencil before and after data reduction.

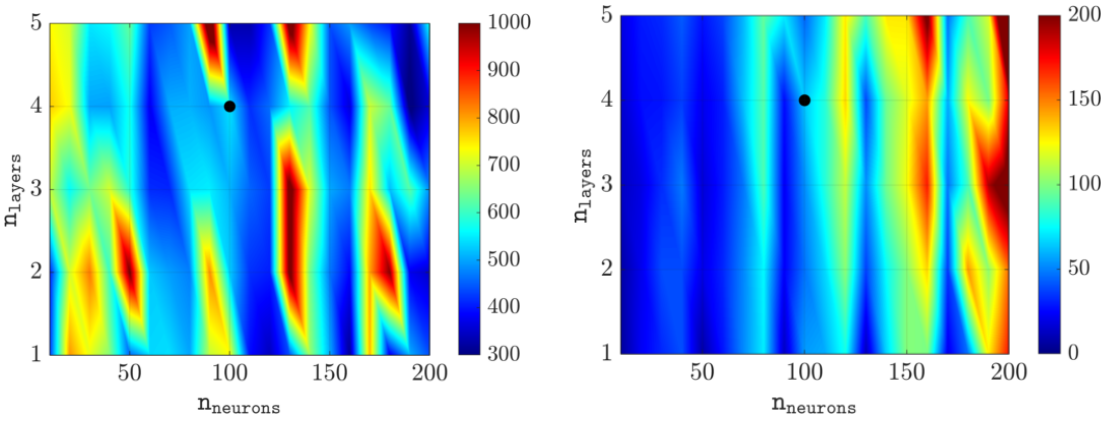


(a) Output 1: u (b) Output 2: v

Figure 4.16: Flow around two circular cylinders: Maximum percentage error of the trained ANN, different number of neurons and hidden layers.

4.4.4 ANN training

To analyse the effect of the hyperparameters of the ANN, training was carried out for different numbers of hidden layers, from 1 to 5, and for different number of neurons in each layer, from 10 to 200 in steps of 10. The error of the ANN, measured as described in Equation (4.6), is shown for both outputs in Figure 4.16. The results show that with four hidden layers and 100 neurons the minimum error, below 2%, is obtained for both outputs. It is worth noting the similar pattern in the error maps for both outputs. Despite a relatively deep network, with four hidden layers, and the relatively large number of neurons, 100, that provide the minimum error, it is important to note that



(a) $\bar{n}_{epochs}$ (b) $\bar{T}_{total}$

Figure 4.17: Flow around two circular cylinders: Mean number of epochs required for training convergence (n_{epochs}) and mean total training time (T_{total}) in seconds for different number of neurons and number of hidden layers.

much shallower networks with fewer neurons provide low errors. For instance, with a single hidden layer and 20 or 30 neurons, the trained ANN provides accurate results, with errors below 4%.

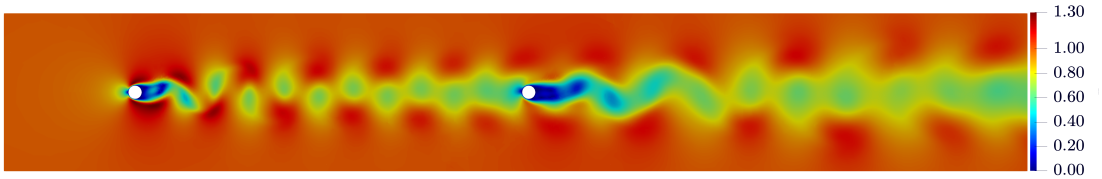
The heat maps in Figure 4.17 reveal the relationship between the architecture of the ANN and the efficiency of training. Each network configuration was trained five times to mitigate the impact of random weight initialisation, with the best performing instance selected. This approach ensures more robust and representative results. Figure (a) depicts the mean number of epochs required for convergence, while Figure (b) illustrates the mean total training time. Both metrics exhibit complex, non-monotonic dependencies on the number of neurons and layers. The architecture selected based on maximum error, consisting of 4 hidden layers and 100 neurons, is situated within a region characterized by a relatively low number of epochs (approximately 500-600) and moderate total training time (around 60-70 seconds). Interestingly, the total training time does not strictly correlate with the number of epochs, indicating that the computational cost per epoch varies significantly across different architectures. This discrepancy is particularly evident for networks with a large number of neurons, where, despite fewer epochs, the total training time increases substantially.

4.4.5 ANN-driven degree adaptivity

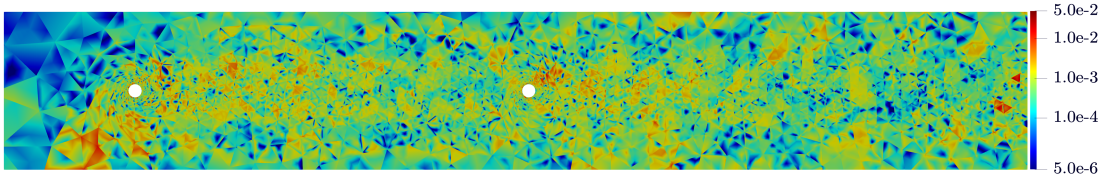
The trained ANN, with four hidden layers and 100 neurons in each layer, is now incorporated within the degree adaptivity process. To this end, at each time step, the error indicator is computed using the current solution at time t^n and using the ANN predicted solution at time t^{n+1}. To compute the error indicator at time t^{n+1}, not only the velocity field is predicted, but also the super-convergent velocity. One of the key aspects of the prediction strategy proposed here is that the same ANN is used to predict both the velocity and the super-convergent velocity.

Figure 4.18 presents the predicted flow field around two circular cylinders in $t = 200$ from $t = 199$, together with associated error maps. The error maps were calculated by taking the absolute error between the predicted solution and the reference solution, presented in Figure 3.11b. The errors are generally low across the domain, exhibiting errors of the order of 10^{-3} to 10^{-4} across the entire domain, indicating a good overall accuracy of the prediction. The error maps focus on the region near the cylinders to highlight the detailed flow features, while the farfield region, though not visible in these plots, exhibits correct and low error levels consistent with the expected behaviour. The error patterns in both x and y components are similar, suggesting that the performance of the model is consistent between different velocity components.

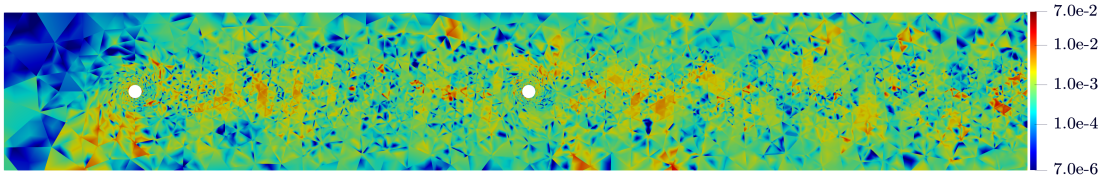
Figure 4.19 illustrates the spatial distribution of of polynomial degrees when using adaptivity, comparing cases with and without the prediction step.In the standard degree adaptivity approach (a), the degree distribution ranges from 1 to 6, reflecting the varying complexity of the local flow features. Higher degree approximations are concentrated near the cylinders and in the near wake regions, where flow features are most intricate. The far wake region shows intermediate degrees, capturing ongoing vortex interactions, while lower degrees are used in the free-stream regions with relatively uniform flow. In contrast, the degree adaptivity with prediction (b) shows a marked increase in higher-degree elements throughout the domain, especially in the wake regions. The transition



(a) Predicted velocity



(b) $|e_u|$

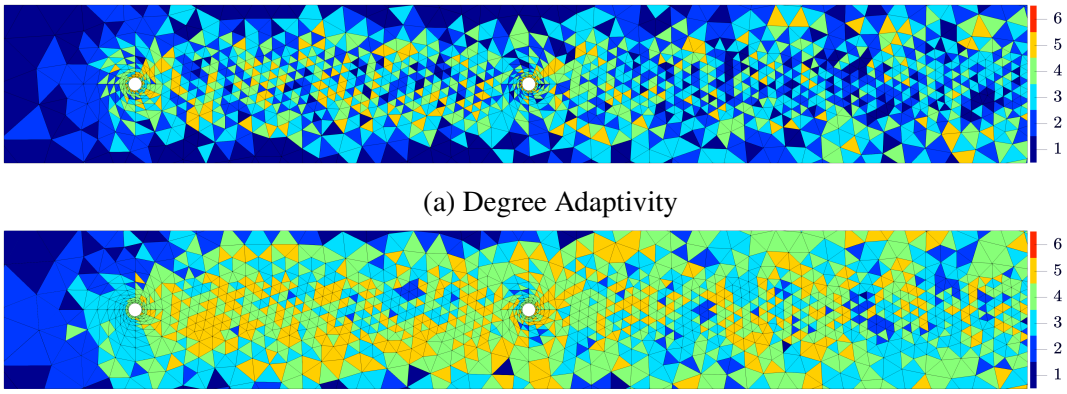


(c) $|e_v|$

Figure 4.18: Flow around two circular cylinders: Predicted velocity fields at $t = 200$ and absolute error maps in logarithmic scale of the velocity in x-direction ($|e_u|$) and the the velocity in y-direction ($|e_v|$) with degree adaptivity.

between degree levels is more gradual, with larger contiguous regions of a similar degree. In particular, higher degrees are applied predictively in areas where complex flow structures are anticipated to develop. Comparing the two approaches reveals that the predictive method results in a more forward-looking degree distribution, increasing approximation orders in regions where the flow is expected to become complex. This strategy is particularly beneficial for unsteady flows, as it increases the polynomial degree before the solution complexity appears in those regions. The standard adaptivity, on the other hand, closely follows the current flow structure, which may be inefficient for rapidly evolving unsteady flows. The differences are most pronounced in the far wake and in areas adjacent to the main vortex street, where the predictive approach anticipates the propagation and development of complex flow structures.

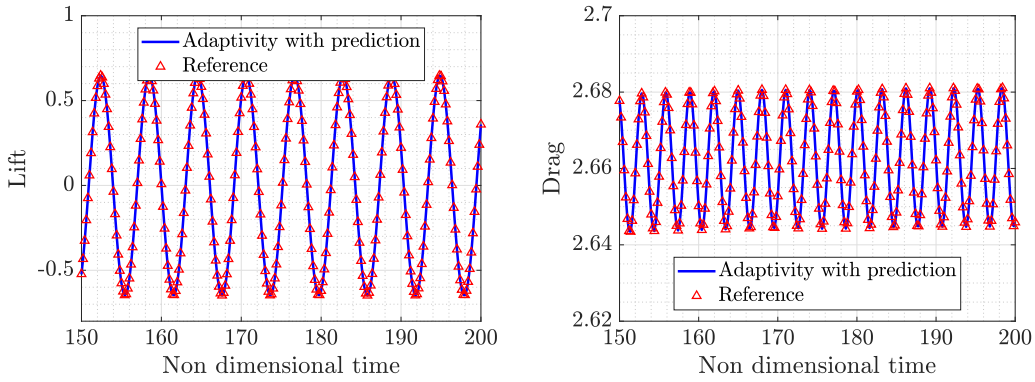
The predictive degree adaptivity demonstrates a significant advantage for unsteady flow simulations. It potentially allows for larger time steps without loss of accuracy, as the solution space is preemptively enriched, and mitigates the accumulation of errors that can occur when the approximation order lags behind the developing flow complexity.



(a) Degree Adaptivity

(b) Degree Adaptivity with Prediction

Figure 4.19: Flow around two circular cylinders: Degree of approximation at $t = 199$ with degree adaptivity and with degree adaptivity including the predicted velocity fields at $t = 200$.



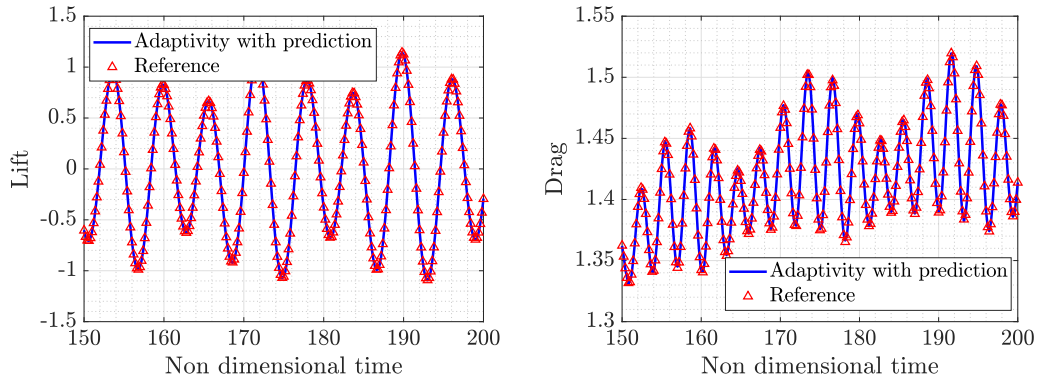
(a) Lift

(b) Drag

Figure 4.20: Flow around two circular cylinders: lift and drag over the first cylinder using degree adaptivity enhanced with prediction compared to the reference solution.

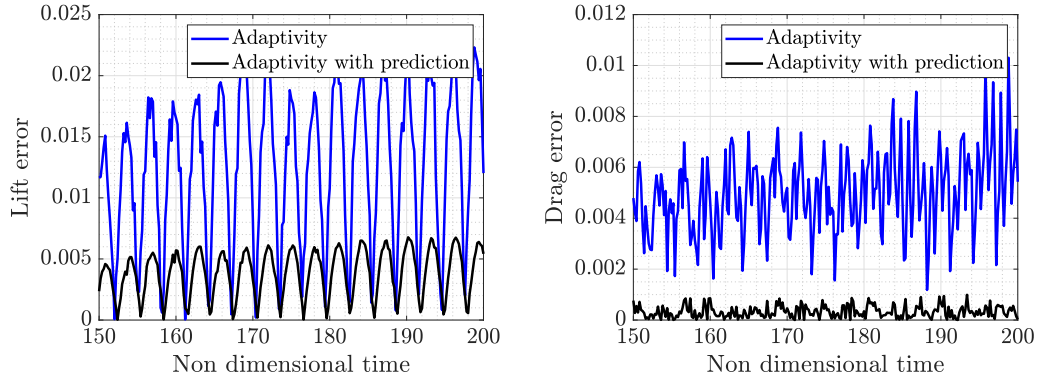
To assess the extra accuracy of the ANN-driven degree adaptive process, Figures 4.20 and 4.21 show the computed lift and drag for the first and second cylinders, respectively. The improvement on the computed drag is clear, with the computed values matching the reference solution for both cylinders.

To quantify the extra accuracy provided by the ANN-driven degree adaptive process, Figures 4.22 and 4.23 display the error of the lift and drag on the first and second cylinders, respectively, as a function of non-dimensional time for the standard degree adaptive strategy and the proposed degree adaptivity aided by the trained ANN. The results show the substantial improvement in accuracy introduced by the prediction of the solution at time t^{n+1} using the trained ANN.



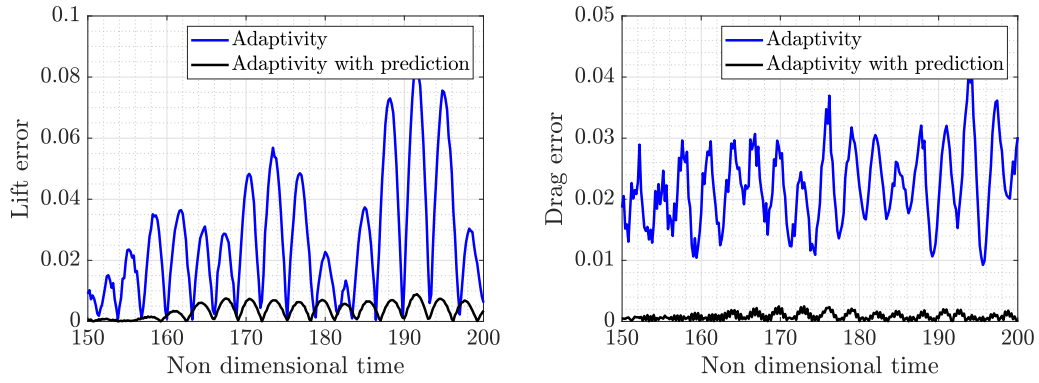
(a) Lift (b) Drag

Figure 4.21: Flow around two circular cylinders: lift and drag over the second cylinder using degree adaptivity enhanced with prediction compared to the reference solution.



(a) Lift (b) Drag

Figure 4.22: Flow around two circular cylinders: error on the lift and drag for the first cylinder as a function of the non-dimensional time.



(a) Lift (b) Drag

Figure 4.23: Flow around two circular cylinders: error on the lift and drag for the second cylinder as a function of the non-dimensional time.

To further quantify the accuracy of the proposed ANN-driven degree adaptive scheme,

Table 4.1 presents the maximum error in the lift and drag forces for both cylinders. The

	Cylinder 1		Cylinder 2	
	Standard adaptivity	**ANN-driven projection**	**Standard adaptivity**	**ANN-driven projection**
Lift error	2.3×10^{-2}	6.8×10^{-3}	8.4×10^{-2}	8.9×10^{-3}
Drag error	1.0×10^{-2}	1.0×10^{-3}	4.1×10^{-2}	2.5×10^{-3}

Table 4.1: Flow around two circular cylinders: maximum error in lift and drag for the two cylinders using the standard adaptivity and the adaptivity with the proposed prediction.

results clearly demonstrate the enhanced accuracy provided by the ANN-driven degree adaptive process. Specifically, the error in the drag force is an order of magnitude more accurate with the proposed method.

Chapter 5

Numerical Examples

This chapter considers the application of the two novel contributions developed in this thesis to problems involving the propagation of gust perturbations over long distances.

Following (Golubev et al., 2009), the gust is modelled using a harmonic perturbation of the velocity field, which is introduced through a source term that influences the momentum equation. This perturbation is designed to create a two-dimensional gust downstream of the source region. Despite there are other alternatives to model the gust (e.g. via the introduction of a boundary condition), the use of a source term is normally preferred to decrease the computational cost, as it avoids refining the path from the far field to the region of interest.

The components of the source term are defined as

$$\boldsymbol{s}(x,y,t) = \begin{cases} \left\{ \begin{array}{l} \beta\ K\ g(x)\ \lambda(y)\ \cos\left(\omega_g t - \beta y - \alpha x_c\right) \\ \\ K\ g'(x)\ \lambda(y)\ \sin\left(\omega_g t - \beta y - \alpha x_c\right) \end{array} \right\} & \text{if } t \in [T^{\texttt{ini}}_{\texttt{gust}}, T^{\texttt{end}}_{\texttt{gust}}] \\ \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{5.1}$$

where (x_c, y_c) represents the centre of a rectangular region of dimensions $a \times b$, within which the gust is generated. The parameters $\alpha = \omega_g / v_\infty$ and $\beta = \alpha \tan\theta$ are the wave numbers corresponding to the sinusoidal gust in the horizontal and vertical directions,

respectively, with θ denoting the angle of propagation of the gust front relative to the x axis, u_∞ the magnitude of the free-stream velocity, and $T^{\texttt{ini}}_{\texttt{gust}}$ and $T^{\texttt{end}}_{\texttt{gust}}$ denote the initial and final non-dimensional time where the source term is active. The constant K, which scales the perturbation, is given by

$$K = \varepsilon_g \, \frac{\alpha \, u_\infty^2 \, (\alpha^2 - \hat{a}^2)}{\hat{a}^2 \, \sqrt{\alpha^2 + \beta^2} \, \sin\left(\dfrac{\omega_g \pi}{u_\infty \hat{a}}\right)}, \tag{5.2}$$

where ε_g represents the gust intensity relative to the mean flow, and $\hat{a}$ defines the width of the rectangular region where the gust is generated, such that $a = 2\pi/\hat{a}$. The functions $g(x)$ and $\lambda(y)$ are chosen to control the spatial distribution of the gust in the x and y directions, respectively, and are defined as

$$g(x) = \begin{cases} \dfrac{1}{2}\left(1 + \cos\left\{\hat{a}\,(x - x_c)\right\}\right), & |x - x_c| \leq \dfrac{\pi}{\hat{a}} \\ 0, & |x - x_c| > \dfrac{\pi}{\hat{a}}, \end{cases} \tag{5.3}$$

$$\lambda(y) = \frac{1}{2}\left\{\tanh\left[3\,(y + y_c)\right] - \tanh\left[3\,(y - y_c)\right]\right\}. \tag{5.4}$$

Here, the function $\lambda(y)$ is selected to ensure a smooth transition in the y direction, creating a compact region with a uniform gust distribution.

5.1 Gust impinging on a NACA0012 aerofoil

The first example, inspired by (Komala-Sheshachala et al., 2020), considers the simulation of a gust impinging on a NACA0012 aerofoil immersed in an incompressible flow at $Re = 1,000$ and aims to demonstrate the benefits of the conservative projection presented in Chapter 3 for a more challenging problem.

The parameters that define the gust are taken as $a = 1$, $b = 4$, $x_c = 1.52$, $y_c = 0$,

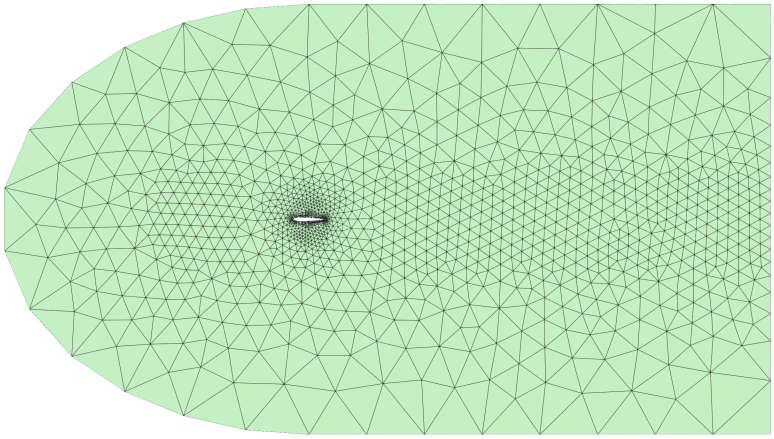


Figure 5.1: Gust impinging on a NACA0012 aerofoil: Unstructured triangular mesh of the whole domain.

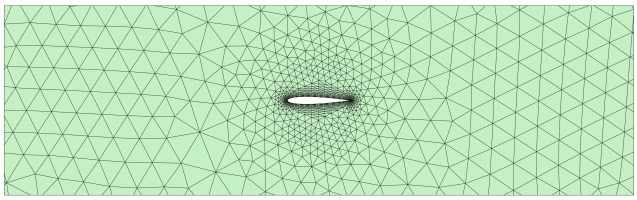
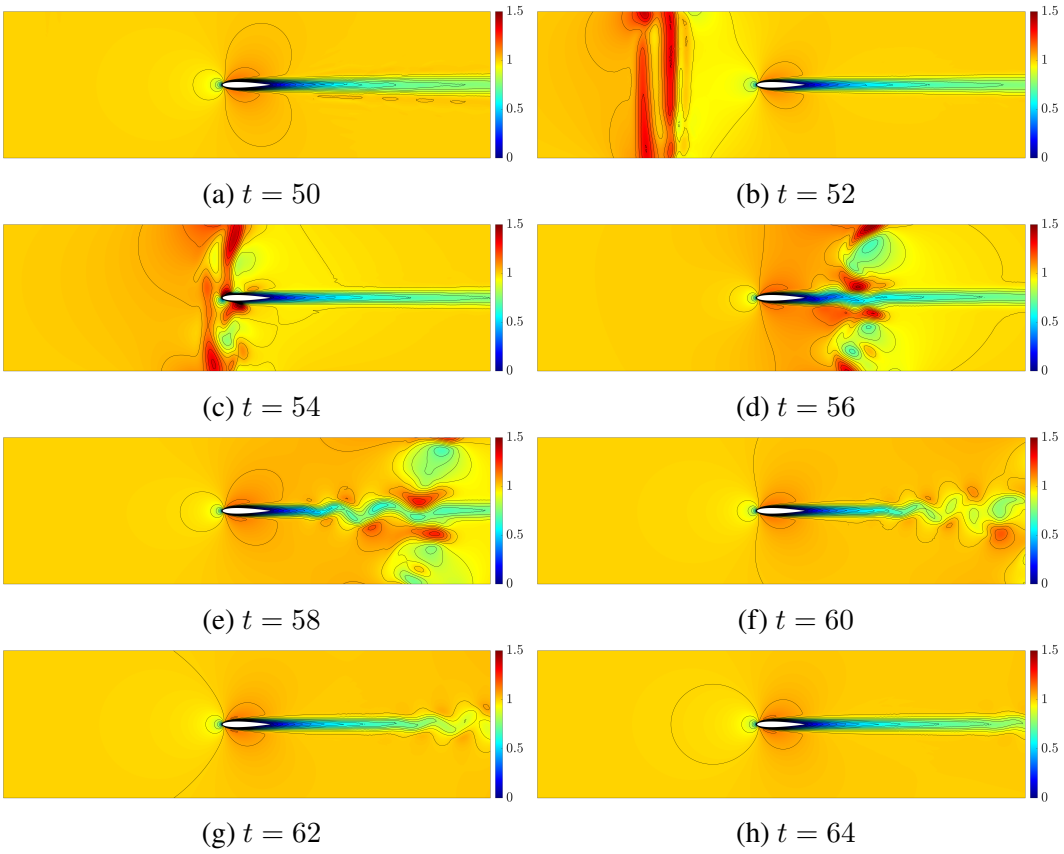


Figure 5.2: Gust impinging on a NACA0012 aerofoil: detail of the unstructured triangular mesh near the aerofoil.

$\alpha = 4\pi$ and $\beta = 0$.

An unstructured mesh of 2,784 triangles is used for this example shown in Figure 5.1. Curved elements are generated near the aerofoil using the elastic analogy presented in (Xie et al., 2013). The size in the normal direction of the first element around the aerofoil is 0.01, and the growth factor in the normal direction is 1.2. Two point sources are introduced to prescribe a mesh size of 0.1 near the leading and trailing edges of the aerofoil, another point source is placed at the centre of the aerofoil to prescribe a size of 0.1 in the vicinity of the aerofoil, whereas a line source with size 0.4 is placed in the path of the gust. A detailed view of the mesh near the aerofoil is shown in Figure 5.2.

Given the complex flow dynamics of this problem, a time step $\Delta t = 0.1$ and the solution is advanced using the ESDIRK46 method until the final time $T = 64$. As commonly done when simulating gust around aerodynamic obstacles (Golubev et al., 2009; Komala-Sheshachala et al., 2020) the initial condition is taken as the steady state solution of the flow around the aerofoil, in this case for $Re = 1,000$. The gust is then



(a) $t = 50$ (b) $t = 52$

(c) $t = 54$ (d) $t = 56$

(e) $t = 58$ (f) $t = 60$

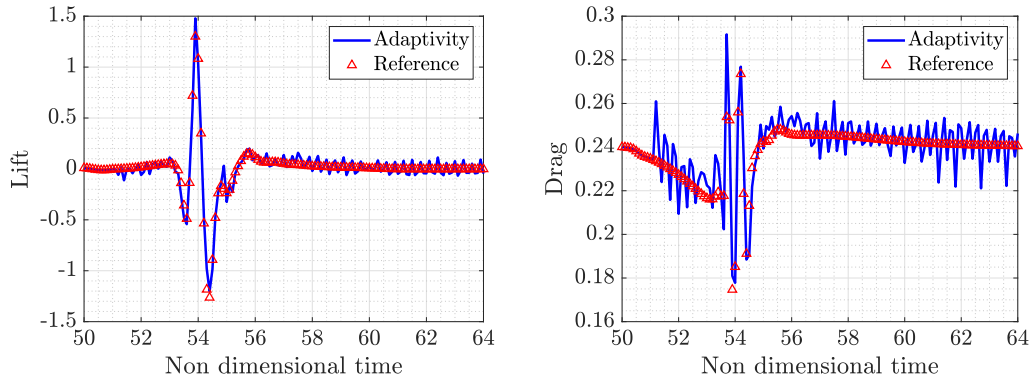
(g) $t = 62$ (h) $t = 64$

Figure 5.3: Gust impinging on a NACA0012 aerofoil: Magnitude of the velocity fields at different instants with a uniform degree of approximation $k = 6$.

introduced via the source term and advanced until the final time, selected so that the gust effect in the aerodynamic forces on the aerofoil disappears.

As in the example involving two cylinders shown in Chapter 3, a reference solution is calculated by using a uniform degree of approximation $k = 6$. The degree of approximation $k = 6$ is selected after performing a convergence study on the fixed mesh of Figure 5.2. The magnitude of the velocity at some selected instants is shown in Figure 5.3, showing the initial steady state solution, the perturbation of the velocity arriving and impinging on the aerofoil, the complex transient effects induced by the gust and the recovery of the steady state solution after the gust effects disappear.

The need for adaptivity in this example is even more obvious than in the previous example because the perturbation of the velocity is very localised and using a high-order approximation in the whole domain is clearly unnecessary. Next, the standard adaptive process and the adaptivity enhanced with the proposed conservative projection



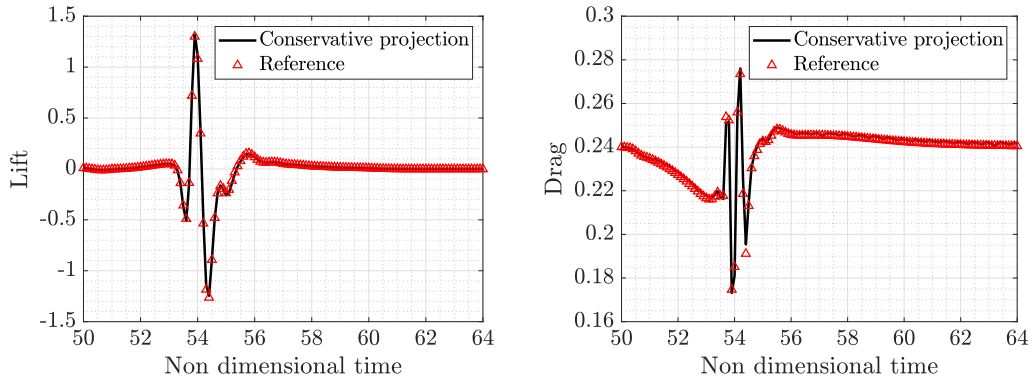
(a) Lift (b) Drag

Figure 5.4: Gust impinging on a NACA0012 aerofoil: lift and drag using degree adaptivity compared to the reference solution.

are considered. To remove the effect of the gust generation, when the source term that generates the gust is active, i.e., for $t \leq 10$, a constant degree of approximation $k = 6$ is used in both cases. After that time, the corresponding adaptive calculation is activated. This ensures that the differences in the adaptive process are not caused by a different representation of the gust. In this example, the desired error of $\varepsilon = 10^{-3}$ is used during the adaptive process.

Figure 5.4 shows the lift and drag on the aerofoil using a standard degree adaptivity, and the results are compared to the reference solution. As in the previous example, the results show non-physical oscillations. The oscillations are more pronounced on the drag but can also be observed on the lift in this example due to the lack of symmetry introduced by the gust. During the transient simulation, a maximum error of 2.3×10^{-1} and 3.8×10^{-2} is observed in the lift and drag, respectively, clearly not providing the accuracy required for this simulation. It is worth noting that from $t = 50$ to $t = 51$ a constant degree of approximation, $k = 6$, is used and as soon as the adaptivity is activated, a strong overshoot in the drag is observed.

When the proposed correction is introduced, an excellent agreement is again observed between the computed lift and drag and the reference solution, as shown in Figure 5.5. For this example, the maximum error in the lift and drag during the whole transient process is 5.4×10^{-2} and 6.2×10^{-3}, respectively, showing the extra accuracy provided



(a) Lift (b) Drag

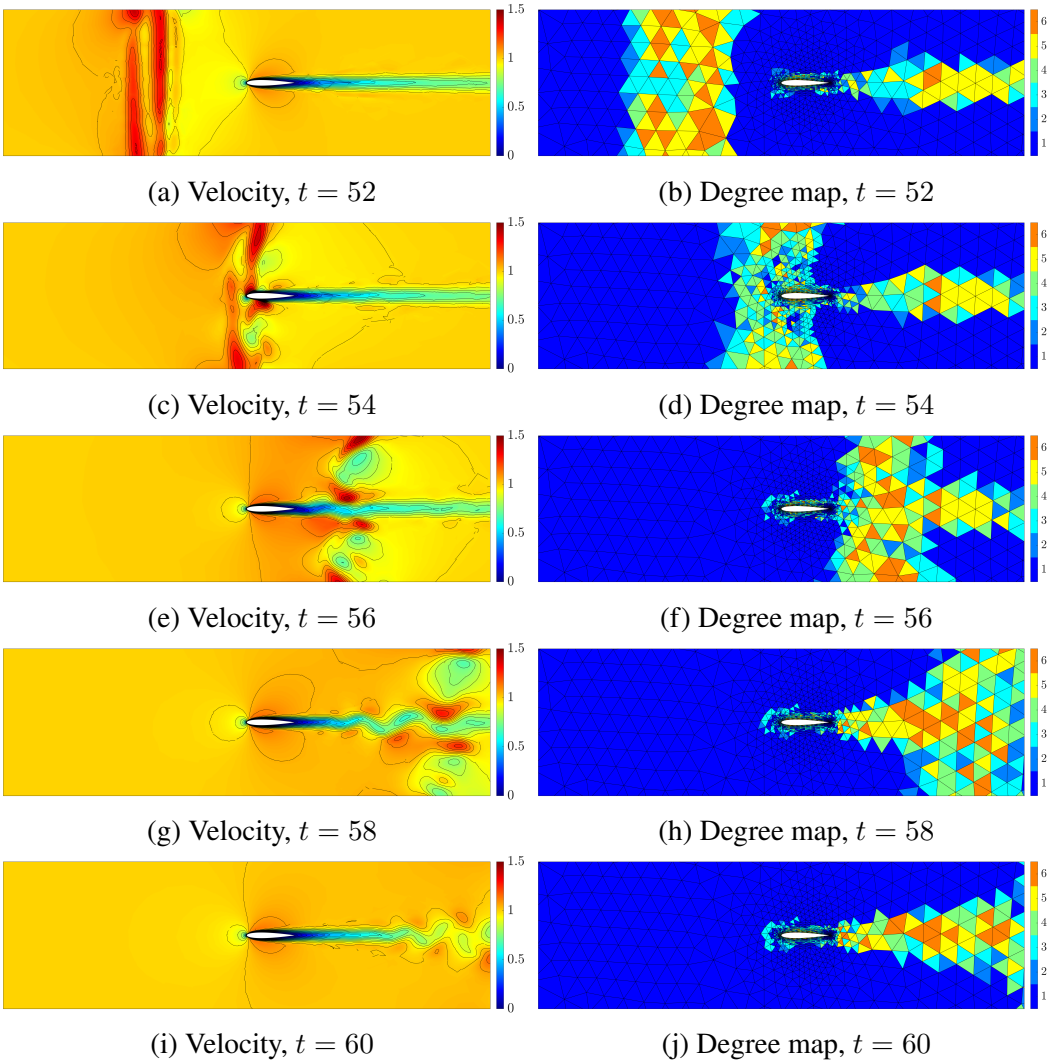
Figure 5.5: Gust impinging on a NACA0012 aerofoil: lift and drag using degree adaptivity and the proposed correction compared to the reference solution.

by the conservative projection of the solution during the adaptive process.

To further quantify the extra accuracy provided by the proposed projection, the $\mathcal{L}_2([51, 64])$ norm of the relative lift and drag error is computed for both adaptive approaches. Without the proposed correction, the errors in lift and drag are 6.3×10^{-2} and 1.4×10^{-3} respectively, whereas when conservative projection is used the errors in lift and drag are more than 40 times lower, namely 1.5×10^{-3} and 2.9×10^{-5}.

To illustrate the ability of the degree adaptive process to accurately capture the complex flow features of this problem, lowering the degree of the elements where accuracy is no longer required, Figure 5.6 shows the magnitude of the velocity and the degree map at some selected instants. Comparing the results with the reference solution of Figure 5.3, it can be observed that the adaptive process captures all the flow features. The degree map clearly reflects the regions where the complexity of the solution requires a higher degree of approximation to provide the desired accuracy.

In this example, the ability to lower the degree of approximation is critical to gain the benefits of a degree adaptive process, without compromising the accuracy. As the gust introduces a localised perturbation of the velocity, without lowering the degree the final degree map shows that a high order polynomial approximation is used in many areas where the flow does not show any feature. The degree map for such an approach is shown in Figure 5.7. To quantify the benefit of the proposed conservative projection,



(a) Velocity, $t = 52$ (b) Degree map, $t = 52$

(c) Velocity, $t = 54$ (d) Degree map, $t = 54$

(e) Velocity, $t = 56$ (f) Degree map, $t = 56$

(g) Velocity, $t = 58$ (h) Degree map, $t = 58$

(i) Velocity, $t = 60$ (j) Degree map, $t = 60$

Figure 5.6: Gust impinging on a NACA0012 aerofoil: Magnitude of the velocity fields (left) and map of the degree of approximation (right) at different instants with the proposed degree adaptive approach.

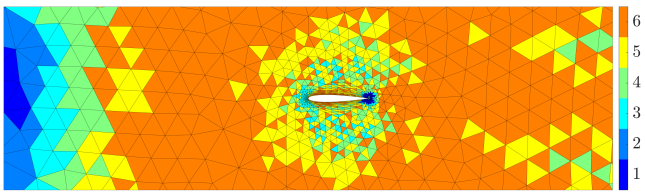


Figure 5.7: Gust impinging on a NACA0012 aerofoil: Map of the degree of approximation at $t = 64$ with an adaptive process not allowing the degree to be lowered.

Figure 5.8 show the number of degrees of freedom of the global problem as a function of the non-dimensional time for the proposed approach and an adaptive process where the degree is not allowed to be decreased during the time marching process. With the proposed projection the number of degrees of freedom at $t = 64$ is 23,518 whereas

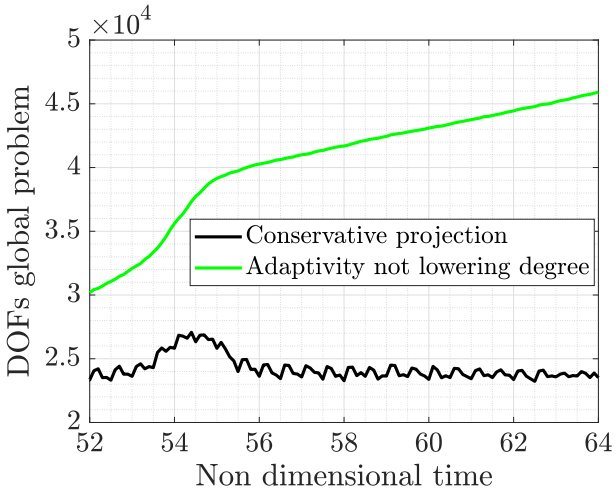


Figure 5.8: Gust impinging on a NACA0012 aerofoil: Number of degrees of freedom of the global problem for two different adaptive approaches.

for the approach not lowering the degree of approximation the number of degrees of freedom at $t = 64$ reaches 45,908. The results with conservative projection show that the most complex dynamics happen at around $t = 54$, which, according to Figure 5.6, is precisely when the gust impinges on the aerofoil. At this point, the number of degrees of freedom of the global problem reaches a maximum and then decreases because the degree of approximation can be lowered in many elements in the vicinity of the aerofoil where the transient effects are no longer relevant.

In terms of computational cost, the simulation with the proposed conservative projection is more than three times faster than the simulation with a uniform degree of approximation $k = 6$. The extra performance compared to the previous example is due to the localised effect of the gust. In this example, the degree adaptive clearly offers a major advantage by introducing high order approximation only where needed.

Table 5.1 provides a detailed analysis of computational costs across different solution strategies. The results demonstrate even more pronounced computational economies compared to the two-cylinder case, with the conservative projection methodology reducing wall-clock time by approximately 69% relative to uniform approximation whilst maintaining solution accuracy. This marked improvement in performance may be attributed to the spatially confined characteristics of the gust phenomenon, enabling

Method	**Total walltime** (h)	**Error indicator** (%)	**Projection time** (%)	**# DOFs** (avg)
Uniform k=6	29.50	-	-	61,864
Conservative projection	9.18	8.5	6.8	25,000

Table 5.1: Gust impinging on a NACA0012 aerofoil: Computational costs.

the adaptive strategy to concentrate high-order approximations solely in the needed regions, as evidenced by the substantial reduction in the mean number of degrees of freedom.

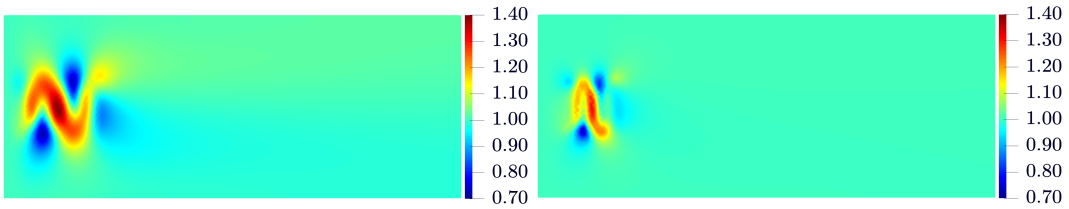
The conservative projection approach achieves these computational savings through two main mechanisms: first, by reducing the total number of degrees of freedom by approximately 44% compared to uniform refinement, and second, by allowing efficient degree reduction in regions where high-order approximation is no longer needed as the gust propagates through the domain. The overhead cost of error estimation and projection operations (approximately 15% of total runtime) is more than compensated by these savings.

5.2 Parametric gust in a free-stream flow

This example considers a parametric analysis in which different sinusoidal gusts are to be propagated in a free-stream flow. The example is used to assess the proposed ANN-driven degree adaptivity for a problem that involves flow parameters. More precisely, the gust is parametrised using the intensity ε_g and width $\hat{a}$. The range of the parameters is taken as $\varepsilon_g \in [0.2, 1]$ and $\hat{a} \in [1, 3]$, whereas the other parameters that characterise the gust are fixed to $x_c = 1.6$ and $y_c = 0$ for the centre of the gust. The angles β and α are set to 0 and 4π, respectively. The initial time of aplication of the gust is $T^{\texttt{ini}}_{\texttt{gust}} = 0$ and the end time is $T^{\texttt{end}}_{\texttt{gust}} = 1.6$, the time step is set to $\Delta t = 0.8$ and the solution is advanced using the ESDIRK816 method until a final time $T = 20$

The computational domain is $\Omega = [0, 20] \times [-4, 4]$ and the setup of the problem is

Figure 5.9: Illustration of the problem setup for the simulation of a gust in a free-stream flow. A sinusoidal gust is generated within the region enclosed by the box of width a and height b, the centre of the box located at (x_c, y_c).



(a) $\varepsilon_g = 0.2$, $\hat{a} = 3$ (b) $\varepsilon_g = 1$, $\hat{a} = 1$

Figure 5.10: Parametric gust in a free-stream flow: velocity field at $t = 1.6$ for different values of the intensity and width.

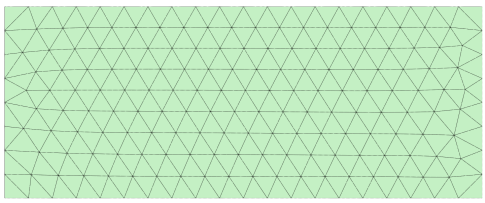
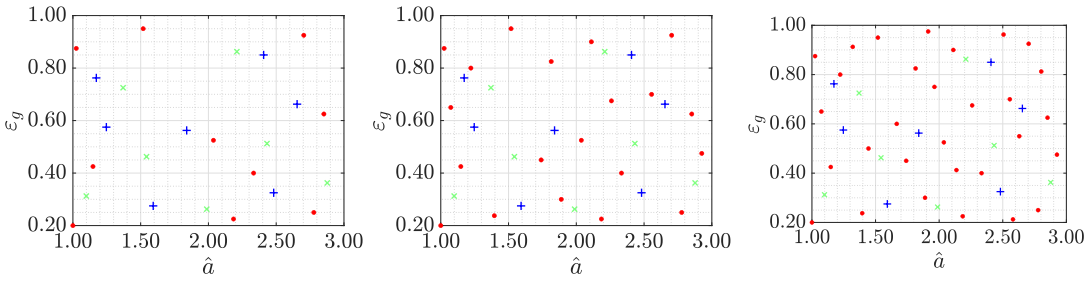


Figure 5.11: Parametric gust in a free-stream flow: unstructured triangular mesh to generate the training data.

illustrated in Figure 5.9.

To illustrate the effect of the parameters, Figure 5.10 shows the velocity field after the gust is introduced for two different choices of the parameters. With $\varepsilon_g = 0.2$ and $\hat{a} = 3$ the gust is relatively wide in the x direction and the intensity is low, when compared to the case with $\varepsilon_g = 1$ and $\hat{a} = 1$ where a higher intensity and lower width produces a more challenging case due to the higher gradients of the velocity field.

An unstructured mesh of 346 triangles, shown in Figure 5.11, is used to generate training data for this example. A uniform element size is chosen to demonstrate the ability of the degree adaptivity to increase the accuracy where required, without any prior mesh adaptation based on the physics of the problem. It is important to emphasise that this mesh is only used to generate the training data and the trained ANN will be



(a) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$ (b) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ (c) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$

Figure 5.12: Gust in a free-stream flow: The parametric space and the three generated datasets, including data for training (in red), validation (in green), and test (in blue).

deployed on a simulation performed on a different, much finer mesh.

5.2.1 Data acquisition and preparation

To generate a good dataset (i.e. a set that provides good coverage of the parametric space), Halton sequencing, as described in Section 4.3.2, is used to sample the parametric space.

One of the objectives of this example is to study the influence of the number of simulations required to gather the data on the accuracy of the ANN predictions. Therefore, a total of 44 sampling points are considered, which means that 44 accurate gust simulations need to be available. The number of simulations is defined by $\mathtt{n}^{\mathtt{sim}}$, and the subindex indicates which set of the data it belongs to, training, validating, or testing. Then three data sets are generated from the available simulations involving $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$, $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ and $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$ simulation cases, respectively, to generate training data. Seven of the remaining 14 simulations are used to generate validation data, and similarly, the last seven cases are used to generate test data unseen by the network during training. The three datasets are shown in Figure 5.12.

After running the 44 simulations, the data acquisition is performed following the procedure described in Section 4.3.2. For this example, the distance of the stencil, computed as described in Section 4.3.1, is $d \approx \boldsymbol{u}_\infty \times \Delta t$.

With the data arranged in input and output arrays, the data preparation process described

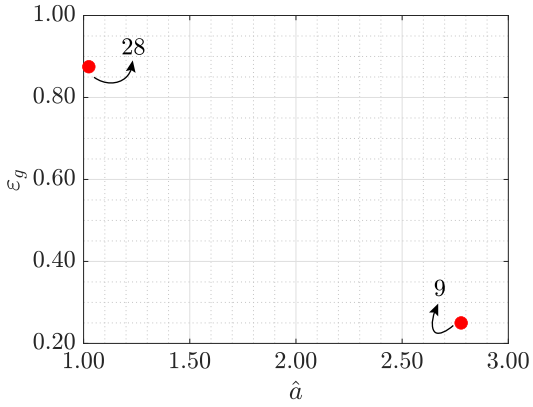


Figure 5.13: Gust in a free-stream flow: Simulations 9 and 28.

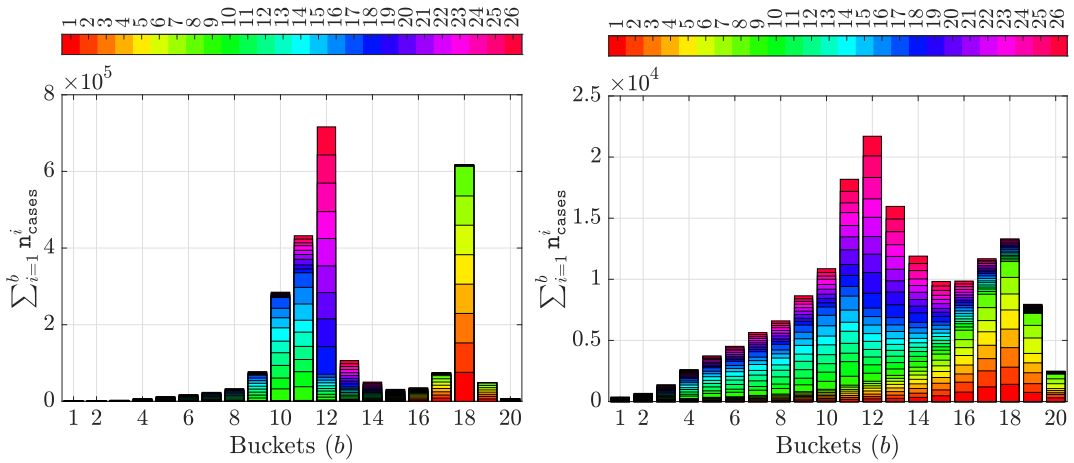
Simulation	p_2		u		v	
	min	max	min	max	min	max
9	0.9225	1.0098	0.8612	1.1337	-0.4669	0.3626
28	0.8635	1.0369	0.7639	1.1880	-0.7126	0.4521

Table 5.2: Gust in a free-stream flow: minimum and maximum values for the pressure on the point 2 and for the velocity on the central point of the stencil, for simulations 9 and 28.

in Section 4.3.3 is followed to remove redundant information and minimise the potential bias in the trained ANN. To illustrate this procedure, two simulations are chosen, shown in Figure 5.13. Simulations 9 and 28 are selected for their representation of extreme cases within the parametric space. Simulation 9 is characterised by $\varepsilon_g = 0.25$ and $\hat{a} = 2.7777$, while simulation 28 is characterised by $\varepsilon_g = 0.875$ and a width of $\hat{a} = 1.0493$.

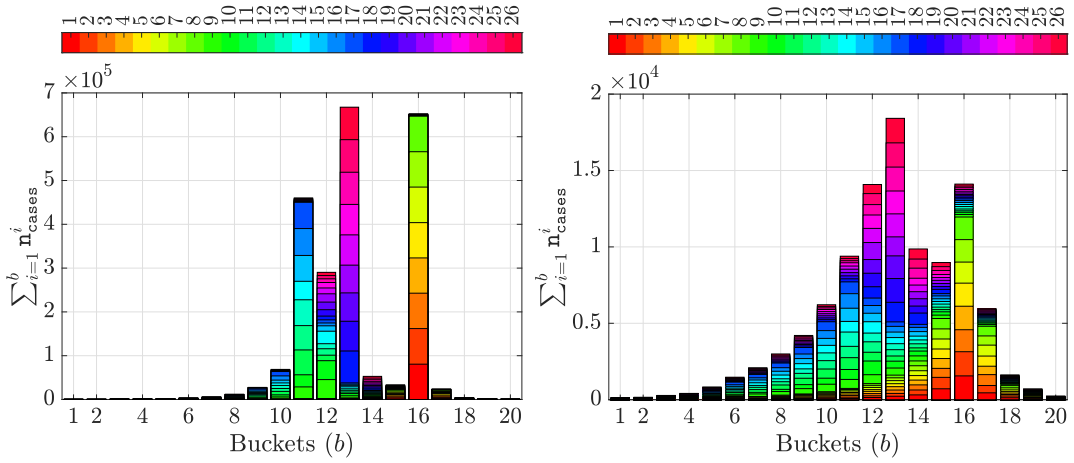
Table 5.2 presents the results of the two simulations (9 and 28), showing minimum and maximum values for pressure (p_2) and velocity components (u and v) at specific points on the input stencil. Simulation 28 exhibits wider ranges for all variables compared to simulation 9. The pressure range is relatively narrow in both simulations, with simulation 28 showing a slightly wider range.

Figure 5.14 and Figure 5.15 show the histograms of the collected data, arranged in $\mathtt{n}_{\mathtt{buckets}} = 20$ buckets before and after the reduction process for simulation 9 and simulation 28 respectively.



(a) Before reduction algorithm (b) After reduction algorithm

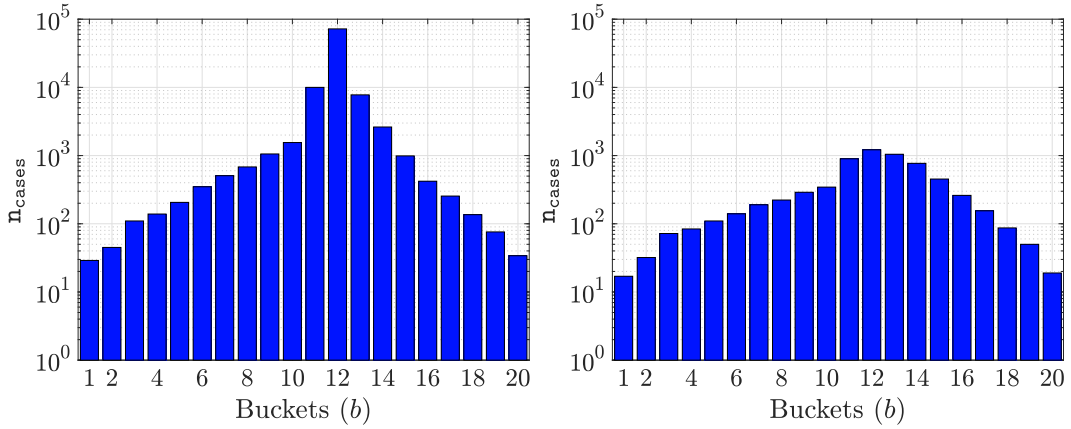
Figure 5.14: Gust in a free-stream flow: Histogram for simulation 9 illustrating the distribution of cases in 20 buckets before and after the application of the data reduction algorithm.



(a) Before reduction algorithm (b) After reduction algorithm

Figure 5.15: Gust in a free-stream flow: Histogram for simulation 28 illustrating the distribution of cases in 20 buckets before and after the application of the data reduction algorithm.

The histograms for Simulations 9 and 28 illustrate the significant impact of the data reduction algorithm on the distribution of cases across 20 buckets. In both simulations, the pre-reduction data show a stark concentration in buckets 11-13, with simulation 9 peaking at approximately 7×10^5 cases in bucket 12, and simulation 28 showing a similar peak of about 6.5×10^5 cases. Post-reduction, the distributions become markedly more uniform. The peak in bucket 12 for simulation 9 reduces to about 2.2×10^4 cases, while for simulation 28 it decreases to approximately 1.8×10^4 cases. In particular, the algorithm increases the relative representation in previously under-represented buckets,



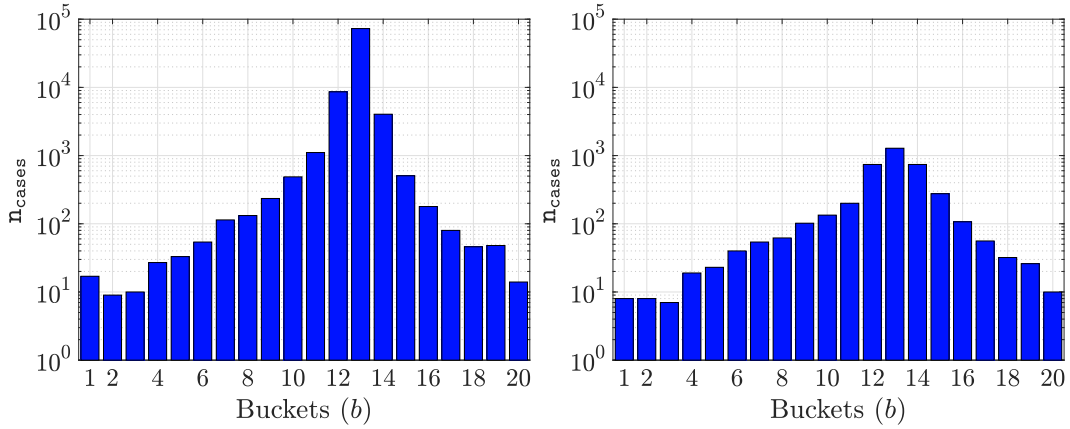
(a) Before reduction algorithm (b) After reduction algorithm

Figure 5.16: Gust in a free-stream flow: Logarithmic histogram for simulation 9 illustrating the distribution of cases in 20 buckets for the vertical velocity at the central point of the stencil before and after data reduction.

especially in the lower and upper ranges. This redistribution effectively "flattens" the distribution while preserving the general shape and key features of the original data. The reduction in the scale on the vertical axis, from 10^5 to 10^4, in both simulations underscores the substantial decrease in the cumulative number of cases. For simulation 9 the total number of cases is reduced from $99,095$ to $6,458$ and, for simulation 28, the total number of cases is reduced from $88,587$ to $3,928$.

The analysis of the impact of the data reduction algorithm on simulations 9 and 28 reveals significant changes in the distribution of cases across various parameters. However, to provide a more focused and insightful examination of the effects of the algorithm, it is highly appropriate to focus on the vertical velocity component (v). This variable exhibits the most pronounced changes in the simulations, making it an ideal candidate for a detailed examination. Figure 5.16 and Figure 5.17 compare the distribution of the data corresponding to the vertical velocity component (v) at the central point of the stencil before and after the reduction process is applied.

The logarithmic histograms for simulations 9 and 28 illustrate the effectiveness of the data reduction algorithm in preserving the overall distribution of vertical velocity cases while significantly reducing the dataset size. In both simulations, the pre-reduction data show a pronounced peak around bucket 12, with simulation 9 reaching nearly 10^5



(a) Before reduction algorithm (b) After reduction algorithm

Figure 5.17: Gust in a free-stream flow: Logarithmic histogram for simulation 28 illustrating the distribution of cases in 20 buckets for the vertical velocity at the central point of the stencil before and after data reduction

cases and simulation 28 exceeding 10^5 cases at this peak. Post-reduction, the general shape of the distributions is maintained, but with a notable flattening effect. The peak in bucket 12 for both simulations is reduced to approximately 10^3 cases, indicating a substantial reduction in data while preserving the central tendency. Importantly, the algorithm maintains representation across all buckets, including the tails of the distributions, which is crucial for capturing extreme events in gust simulations. The reduction process appears to be more aggressive in the high-frequency buckets while being more conservative in the low-frequency ones, effectively balancing data reduction with the preservation of rare events.

Analysis of histograms for simulations 9 and 28 reveals the significant impact of the data reduction algorithm on the distribution of cases in 20 buckets. However, it is crucial to note that the intervals for these simulations differ substantially. To address this disparity, each simulation undergoes an individual reduction process. This tailored reduction preserves the unique features of each dataset while significantly decreasing the total number of cases. Following the individual reduction of each simulation, the data are combined for training the ANN by combining the data points and adding the two parameters at the end of each case. By including these parameters, the combined dataset retains essential information about the specific characteristics of each original

$\mathtt{n}^{\mathtt{sim}}$	$\mathtt{n}_{\mathtt{Tr}}$	$\mathtt{n}_{\mathtt{Val}}$	$\mathtt{n}_{\mathtt{Test}}$
10	49,029	37,789	34,563
20	98,492	37,789	34,563
30	148,046	37,789	34,563

Table 5.3: Gust in a free-stream flow: number of cases for training, testing, and validation utilising 10, 20, and 30 simulations for training.

simulation. This approach ensures that, despite the reduction in data volume, the fundamental differences between simulations are preserved and can be accounted for in subsequent analyses.

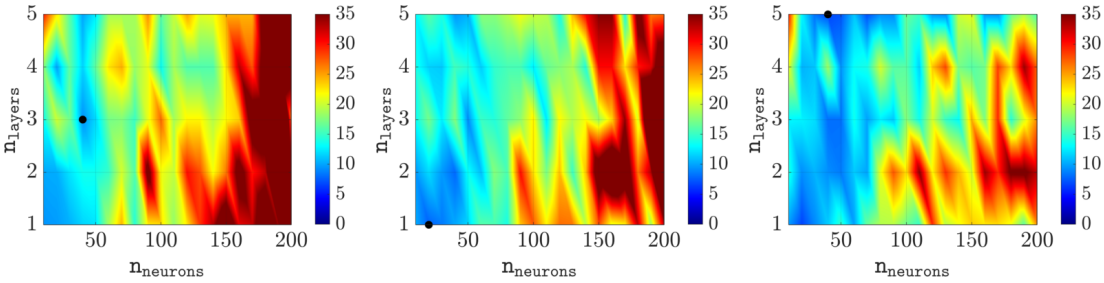
Finally, the number of cases for training, testing and validating is shown in Table 5.3. It is worth noting that, following common practice, the number of validation and test cases is fixed and it is independent on the amount of training cases.

Next, a series of numerical studies are presented to assess the influence of the amount of data and the reduction process on the accuracy of the ANN. These studies also explore the influence of the distance used in the stencil and the use of other types of stencil that lead to different ANN architectures.

5.2.2 Influence of the dataset

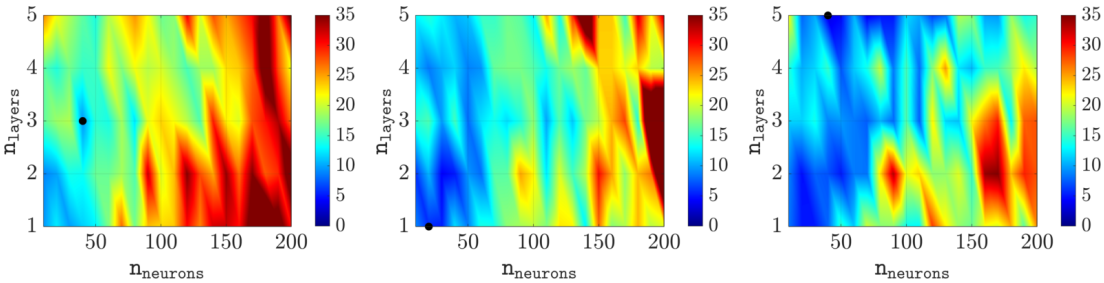
Using the three datasets shown in Figure 5.12, ANNs are trained. The process follows the same rationale as described in Section 4.4.4, that is, hyperparameter tuning is performed using a simple grid search, by trying different number of hidden layers and different number of neurons.

Figure 5.18 and Figure 5.19 present the percentage error in the predicted horizontal velocity (u) and vertical velocity (v) with varying architectures. The results are displayed as 2D contour plots for different numbers of training simulations $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$, $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ and $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$. The horizontal axis represents the number of neurons per hidden layer, while the vertical axis represents the number of hidden layers in the ANN. As the number of training simulations increases from 10 to 30, the overall error tends



(a) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$ (b) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ (c) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$

Figure 5.18: Gust in a free-stream flow: Maximum percentage error of the trained ANN for the horizontal velocity (u), different number of neurons and hidden layers.

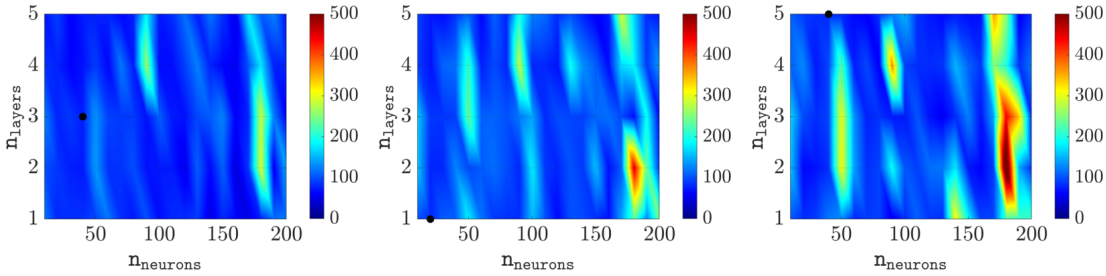


(a) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$ (b) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ (c) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$

Figure 5.19: Gust in a free-stream flow: Maximum percentage error of the trained ANN for the vertical velocity (v), different number of neurons and hidden layers.

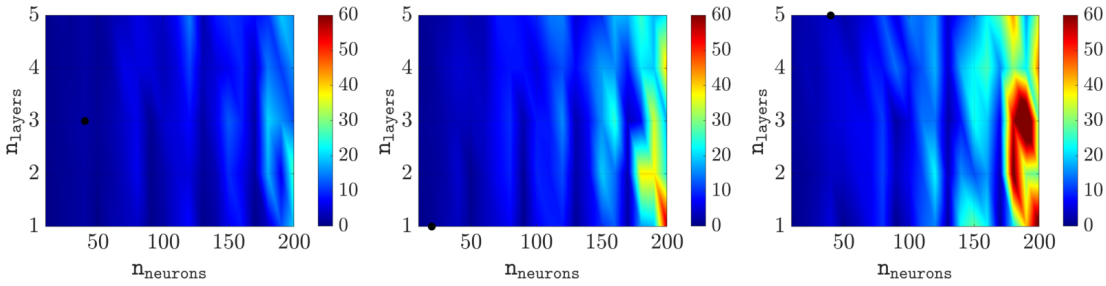
to decrease, as evidenced by the expansion of blue regions and the reduction of red regions from plot (a) to (c). It is particularly remarkable that ANNs with a single layer and a very low number of neurons can provide accurate results, whereas introducing more non-linearity in the ANN (i.e. increasing the number of layers) and increasing the number of neurons tend to provide less accurate ANN.

The observation that increasing ANN complexity through additional layers and neurons may lead to reduced accuracy presents an intriguing paradox in machine learning. While it might seem logical that greater network capacity should enhance representational capabilities, empirical evidence indicates that more complex architectures often result in degraded performance. This phenomenon can be attributed to several factors: the increased difficulty in optimising larger parameter spaces, enhanced susceptibility to overfitting, and challenges in gradient propagation through deeper networks. These challenges manifest in the training phase, where more complex networks may struggle to converge to optimal solutions. Moreover, the computational overhead and increased



(a) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$ (b) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ (c) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$

Figure 5.20: Gust in a free-stream flow: Mean number of epochs required for training convergence ($\bar{\mathtt{n}}_{\mathtt{epochs}}$), different number of neurons and hidden layers.



(a) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 10$ (b) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 20$ (c) $\mathtt{n}_{\mathtt{Tr}}^{\mathtt{sim}} = 30$

Figure 5.21: Gust in a free-stream flow: Mean total training time ($\bar{T}_{\mathtt{total}}$) in minutes, different number of neurons and hidden layers.

training time associated with more complex architectures may not justify the marginal improvements—or indeed decrements—in performance. This suggests that simpler neural network architectures may actually be more effective for many applications, as evidenced by this example where simpler architectures consistently achieve lower error rates.

Figure 5.20 shows the mean number of epochs required for the convergence of the training, while Figure 5.21 displays the mean total training time in minutes.

Both figures present results for different network architectures (varying numbers of neurons and hidden layers) and training dataset sizes (10, 20, and 30 simulations). As the number of simulations increases, larger networks (more neurons and layers) tend to require more epochs to converge, as indicated by the increasing red areas in the upper right corners. Simpler networks (fewer neurons and layers) maintain relatively consistent convergence epochs across different dataset sizes. Training time generally increases with network complexity and dataset size, as expected. However, it is worth

Process	**Time**
Training simulations	20 minutes
Data preparation	3.5 minutes
Neural network training	6 minutes
Total time	**29.5 minutes**

Table 5.4: Computational time breakdown for the selected artificial neural network.

noting that the time for training the ANN is very low, especially for those ANN that provide lower errors.

As illustrated in Figure 5.21, the training process requires only a few minutes. However, the total computational cost must account for the simulation times of the training cases. Each training simulation requires less than 20 minutes of computational time, with all cases executed simultaneously. This parallel implementation provides efficient generation of the training dataset, whilst maintaining modest computational requirements compared to traditional full-domain simulations.

To provide a comprehensive overview of the computational requirements, Table 5.4 presents the breakdown of processing times for the selected neural network configuration.

For each dataset, the ANN that provides the best performance is selected and Figure 5.22 shows the maximum error, in percentage, of the two components of the velocity field as a function of the number of simulations used to collect data for network training. The results show that even with only 10 simulations the error is approximately 10% for both components. Increasing the size of the dataset provides a reduction of the error for both components, reaching an error of 7% and 3% for the horizontal and vertical components of the velocity, respectively, when using the 30 available simulations.

A numerical parameter introduced in the data preparation stage is the number of buckets used for the data reduction process. This parameter controls the amount of data selected to train the ANN and the amount of redundant data present in the dataset. To quantify the effect of the number of buckets in the size of the dataset, Table 5.5 reports the

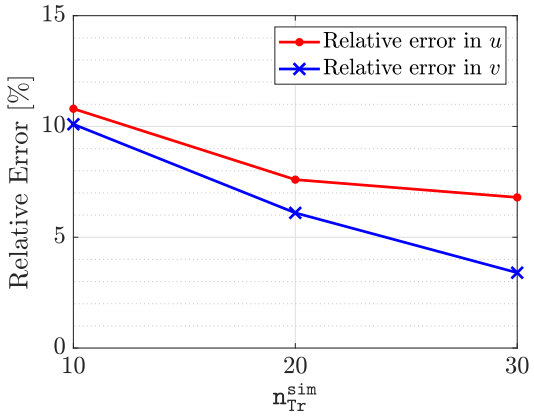


Figure 5.22: Gust in a free-stream flow: maximum error as a function of the number of simulations used to collect data for training the network (n_{Tr}^{sim}).

$n_{buckets}$	n_{Tr}
5	11,112
10	52,517
20	148,046
40	314,731
80	535,586
160	766,299

Table 5.5: Gust in a free-stream flow: number of resultant training cases (n_{Tr}) when varying the number of buckets used for the reduction procedure ($n_{buckets}$).

resulting number of training cases for a different number of buckets.

To analyse the effect of the data reduction algorithm proposed in Section 4.3.3, Figure 5.23 shows the maximum error as a function of the number of buckets used for the reduction. Figure 5.23 reveals that, in this example, the optimal number of buckets is 20, which yields to the lowest error rates for the horizontal (u) and vertical (v) velocity components. The increase in error rates beyond 20 buckets, despite the larger training datasets, suggests that data quality and representation are more crucial than quantity alone. This behaviour may indicate overfitting with larger datasets, where the model learns specific patterns that do not generalise well. This experiment indicates that simply increasing the data set by increasing the number of buckets does not necessarily improve the predictive capability of the ANN. This is attributed to the fact that adding more data in this example produces a bias that results in a higher error in the ANN predictions. Specifically, exceeding the optimal number of buckets introduces redun-

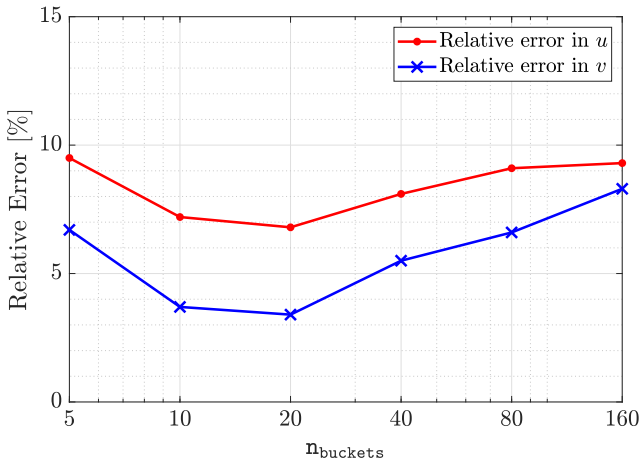


Figure 5.23: Gust in a free-stream flow: maximum relative error as a function of the number of buckets ($n_{buckets}$) chosen in the reduction process.

dant sampling points that add noise rather than new information, thereby degrading the ability of the ANN to capture the underlying flow physics.

5.2.3 Influence of the stencil distance

As mentioned in Section 4.3.1, the definition of the distance used to define the stencil employed to gather the data from a simulation is purely based on physical considerations.

To study the effect of varying the distance, Figure 5.24 shows the relative error in predicting the velocity components as a function of the stencil distance d. The time step for these simulations is set to $\Delta t = 0.80$, and the free-stream velocity is $u_\infty = 1$.

As the stencil distance increases, the error generally decreases for both velocity components, indicating improved prediction accuracy. In particular, the optimal stencil distance aligns with the product of the time step and the free-stream velocity, that is, $d = 0.80$ for this example. However, further increasing the stencil distance to $2u_\infty \times \Delta t$ results in a sharp increase in error for both components. This outcome is expected in the context of a gust propagating in a free-stream flow, where the velocity field changes are best captured when the stencil distance corresponds to the distance a fluid particle travels during one time step. The results emphasise the crucial importance of choosing

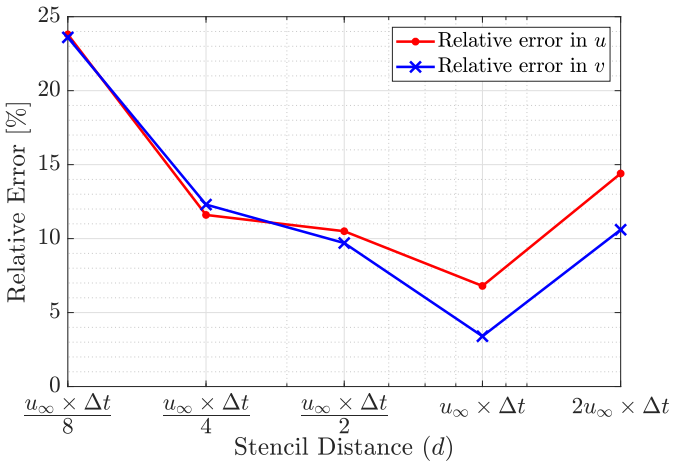


Figure 5.24: Gust in a free-stream flow: maximum error as a function of the stencil distance (d).

an appropriate stencil distance relative to the time step and the characteristic velocity of the flow.

5.2.4 Influence of using several time steps as inputs of the ANN

All the experiments done so far consider, as inputs of the ANN, the values of pressure and velocity at the stencil points and at time t^n. Numerical experiments were conducted to explore the possibility of enhancing the accuracy of the predictions by using information from previous time steps at the stencil points and different stencil distances.

Figures 5.25 and 5.26 represent two new stencil designs to compare with the original one presented in Figure 4.3 and used in all previous examples. It is worth noting that the number and position of the points with respect to the central mesh node do not change with respect to the original stencil used previously. The difference lies in the information stored at the stencil points. The original stencil considered the velocity at the stencil nodes at time t^n, while the stencil in Figure 5.25 considers the velocity at times t^n and t^{n-1}. Similarly, the stencil of Figure 5.27 considers the velocity at the stencil nodes at times t^n, t^{n-1} and t^{n-2}. In all cases, pressure is only considered at time t^n, mainly motivated by the fact that the time derivative of the velocity appears in the

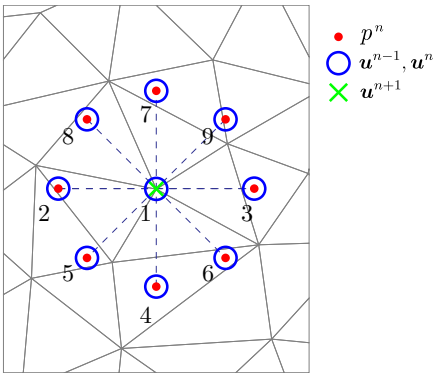


Figure 5.25: Schematic representation of the inputs and outputs for an alternative ANN architecture. The input stencil includes the velocity in two time steps $\boldsymbol{u}^{n-1}$, $\boldsymbol{u}^n$, and pressure p^n. Output consisting of $\boldsymbol{u}^{n+1}$ at the central mesh node.

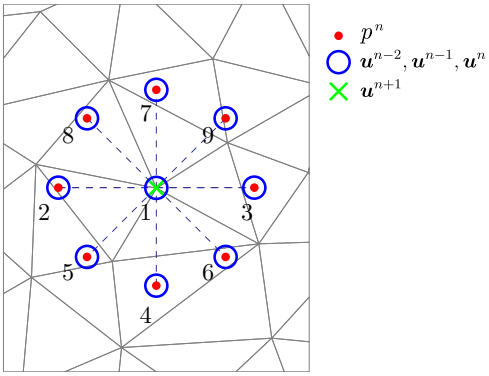


Figure 5.26: Schematic representation of the inputs and outputs for an alternative ANN architecture. The input stencil includes the velocity in three time steps $\boldsymbol{u}^{n-2}$, $\boldsymbol{u}^{n-1}$, $\boldsymbol{u}^n$, and pressure p^n. Output consisting of $\boldsymbol{u}^{n+1}$ at the central mesh node.

Navier-Stokes equations, but not the time derivative of the pressure.

Accounting for extra information in the stencil points obviously leads to ANN with an increased number of inputs. The stencil in Figure 5.25 results in an ANN with $44 + \mathtt{n}_{\mathtt{param}}$ inputs, whereas the stencil in Figure 5.26 results in an ANN with $62 + \mathtt{n}_{\mathtt{param}}$ inputs, compared to the original stencil which had $26 + \mathtt{n}_{\mathtt{param}}$ inputs.

To evaluate the performance of the ANN created from the different stencils, the relative error in the predicted velocity is compared in Figure 5.27 for each stencil and for different values of the stencil distance.

The results show that the inclusion of additional historical time steps did not produce substantial improvements in accuracy that would justify increasing the size of the input layer and consequently the training time. The error trends for the three cases

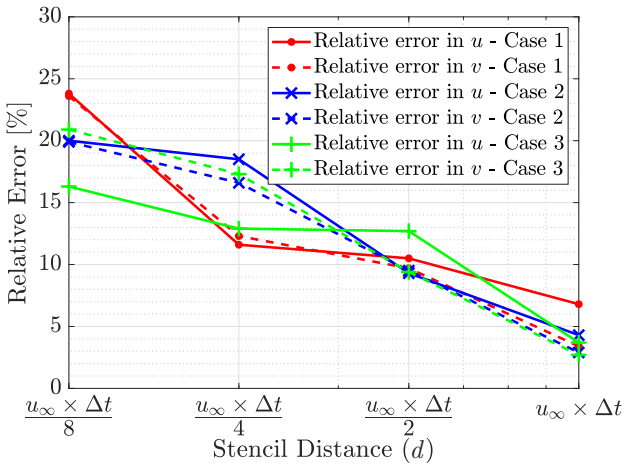


Figure 5.27: Relative error of the predicted velocity as a function of the stencil distance (d) for the stencils that uses the velocity at time t^n (Case 1), at times t^n and t^{n-1} (Case 2) and at times t^n, t^{n-1} and t^{n-2} (Case 3).

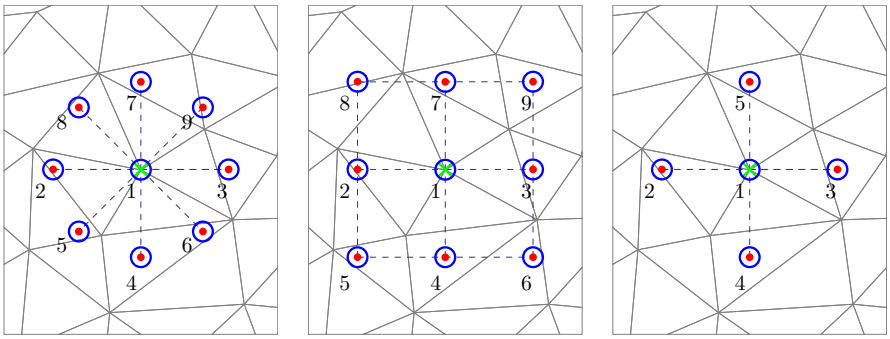
exhibit broadly similar characteristics, with errors decreasing as the stencil distance approaches the value defined using physical considerations, that is, $d = u_\infty \Delta t$. The lack of significant enhancement from incorporating more time history may be attributed to the nature of the flow physics being modelled. For the problem under consideration, it appears that spatial correlations exert a stronger influence on local flow evolution compared to temporal correlations over the timescales examined.

5.2.5 Influence of the stencil geometry

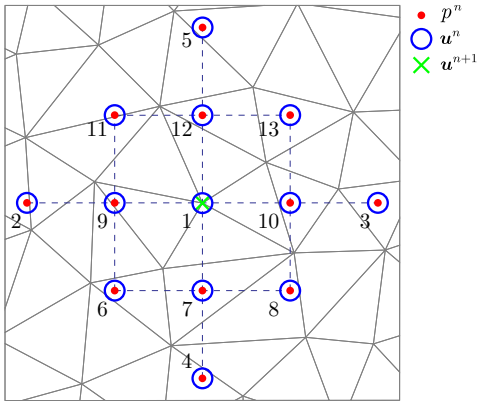
The last numerical experiment involves a study to determine the potential benefit of varying the geometry of the stencil considered. Four configurations, shown in Figure 5.28, are considered for this study.

Following the notation previously introduced in Section 4.3 red dots denote pressure sampling points, blue circles represent horizontal velocity sampling points, and green crosses indicate vertical velocity sampling points.

The first stencil is the one originally introduced in Section 4.3, where eight nodes are placed in an imaginary circle of radius d centred at a mesh node. The second stencil maintains the number of stencil points of the original one, but changes the position,



(a) Type 1 (b) Type 2 (c) Type 3



(d) Type 4

Figure 5.28: Gust in a free-stream flow: schematic representation of the four stencils used to assess the influence of the geometry of the stencil.

leading to the arrangement of a Cartesian grid. The third option is inspired by a finite difference stencil and involves only four points around the central mesh node. The last option is inspired by the stencil that is used in the semi-implicit method for pressure linked equations (SIMPLE) algorithm (Patankar and Spalding, 1983), traditionally employed by incompressible finite volume solvers.

The first two stencils lead to an ANN architecture with $26 + \mathtt{n_{param}}$ inputs, whereas the second and third involve $14 + \mathtt{n_{param}}$ and $38 + \mathtt{n_{param}}$ inputs. Therefore, this study enables us to assess how maintaining the number of inputs but varying the geometry of the stencil influences the accuracy and also how increasing or decreasing the number of inputs influences the predictive capability of the ANN.

Table 5.6 presents a comparative analysis of the maximum relative errors for different stencil types in predicting velocity components during gust propagation in a free stream flow.

	Relative error [%]	
Stencil type	u	v
1	6.8	3.4
2	5.9	4.1
3	9.2	7.5
4	6.6	4.3

Table 5.6: Gust in a free-stream flow: maximum relative errors for different stencil geometries.

Given the results of the previous study, all simulations were performed with a stencil distance $d = u_\infty \Delta t$.

Comparing the accuracy of the first two stencils, it seems that, for a given number of inputs, there is no clear advantage on a particular geometric configuration. Both provide a similar accuracy for both horizontal and vertical velocity components. The third stencil, which involves the minimum number of inputs, leads to an ANN that exhibits the highest error in the predictions, with a maximum error of 9.2% in the horizontal velocity predictions. Finally, the last stencil, which involves the maximum number of inputs, provides a similar accuracy compared to the first two stencils, showing that the addition of extra information does not involve better predictions.

The last two stencils are not considered suitable due to the larger errors or extra complexity and data required, respectively. The first two stencils seem to offer the right balance between computational complexity and accuracy, and only the first one is considered next to demonstrate its use in a degree adaptive process.

5.2.6 ANN-driven degree adaptivity

To conclude this example, the trained ANN is incorporated into the degree adaptive process to simulate the propagation of gust in a free-stream flow. Using the knowledge acquired in the previous numerical experiments, the stencil of Figure 5.28(a) is considered, with a distance $d = u_\infty \Delta t$ and only using the velocity and pressure at time t^n as inputs.

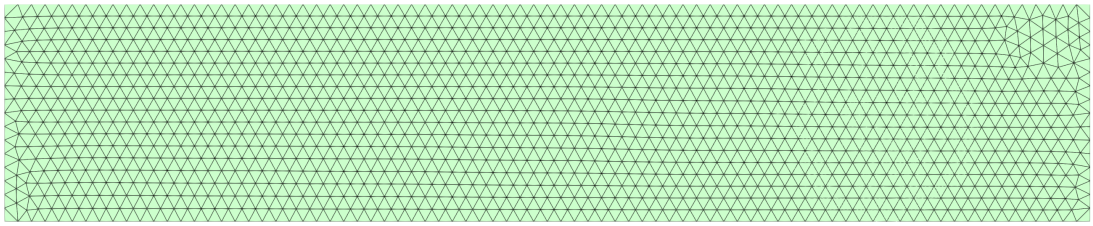


Figure 5.29: Parametric gust in a free-stream flow: unstructured triangular mesh to use for adaptivity.

The ANN is trained with the data gathered from 30 simulations, and the data reduction is performed using 20 buckets. The chosen architecture, based on the results seen in Figure 5.18c and Figure 5.19c, is 5 hidden layers and 40 neurons. This leads to a maximum error in u of 6.80% and in v of 3.40%.

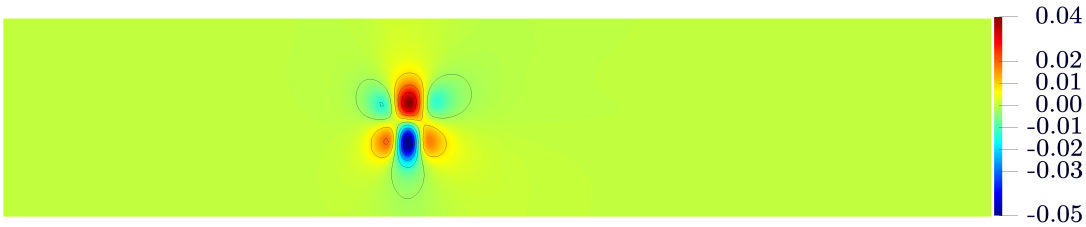
The adaptivity will be evaluated in a significantly larger domain than the one used during training. The computational domain is $\Omega = [0, 80] \times [-8, 8]$. This decision was taken to examine adaptivity across an expanded temporal range and to ascertain whether an ANN, initially trained in one domain, can be effectively generalised to another. An unstructured mesh of 2,858 triangles, shown in Figure 5.29, is used for this example. A uniform element size is chosen to demonstrate the ability of the degree adaptivity to increase the accuracy where required, without any prior mesh adaptation based on the physics of the problem.

An unseen scenario for the ANN is considered, corresponding to $\varepsilon_g = 0.75$ and $\hat{a} = 1.10$. The reference solution, computed in the new domain using a uniform degree of approximation $k = 5$, is displayed in Figure 5.30 for several times. The results illustrate the challenge of propagating a localised perturbation over long distances, where the use of high-order elements is particularly attractive due to the low dissipation and dispersion properties of these schemes.

The different snapshots also clearly suggest that a degree adaptive process is crucial to ensure an efficient use of the computational resources as the velocity field is almost constant in the majority of the domain, with large gradients localised in small regions and travelling across the domain.

(a) $t = 1.6$

(b) $t = 16$



(c) $t = 32$

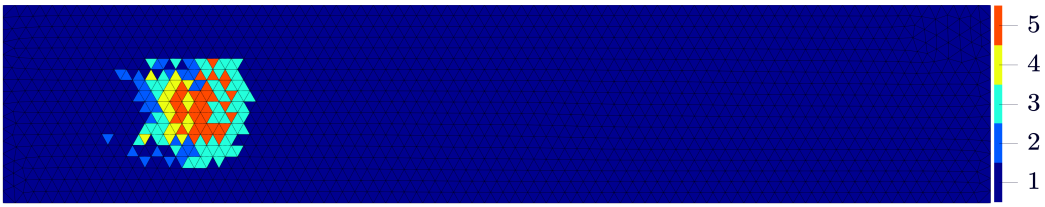
(d) $t = 64$

Figure 5.30: Gust in a free-stream flow: vertical velocity field (v) at different instants with a uniform degree of approximation $k = 5$.

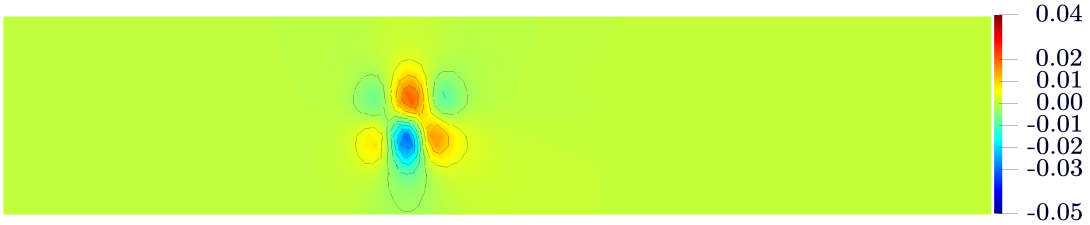
Next, the degree adaptive process enhanced with the conservative projection proposed in Chapter 3 but without using an ANN to predict the solution at the next time step is considered.

Figure 5.31 shows the vertical velocity field at different instants and the corresponding degree map. The results show that the degree adaptive process targets the required error using the snapshot of the velocity at t^n. However, it can be clearly observed that ahead of the gust a linear approximation is used in all elements because there is no flow feature that requires a higher order. As the solution propagates, the mesh is therefore not adapted to properly capture the gust disturbance that arrives at a future time step.

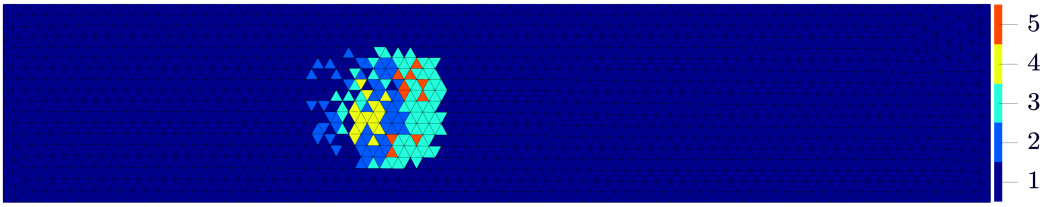
(a) Vertical velocity field (v) at $t = 16$



(b) Degree of approximation at $t = 16$

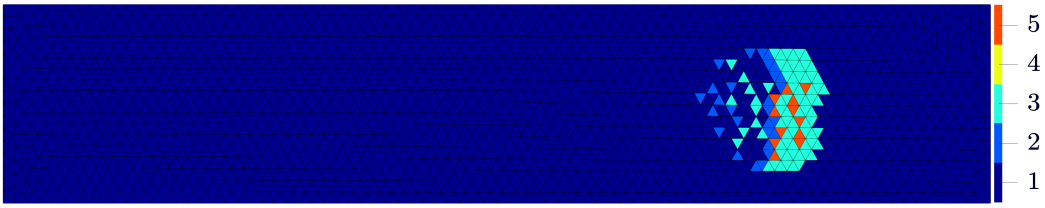


(c) Vertical velocity field (v) at $t = 32$



(d) Degree of approximation at $t = 32$

(e) Vertical velocity field (v) at $t = 64$



(f) Degree of approximation at $t = 64$

Figure 5.31: Gust in a free-stream flow: vertical velocity field (v) and degree of approximation at different instants with degree adaptivity.

This effect leads to dissipation and dispersion of the solution, as typically seen in simulations with low order elements, which is already visually observed at $t = 32$.

The issue is exacerbated as the solution evolves in time because the more dissipated the solution, the lower the degree of approximation that the adaptivity process requires. The final snapshot corresponding to $t = 64$ clearly shows that the adaptive process has not been able to capture the solution because the velocity perturbations have lost the symmetric pattern exhibited by the reference solution and the intensity is much lower.

Next, the simulation is repeated with the ANN utilised in the adaptive process to predict the solution at t^n and use this information to perform the degree adaptation. Figure 5.32 shows the vertical velocity field at different instants and the corresponding degree map with the proposed ANN-driven degree adaptive process.

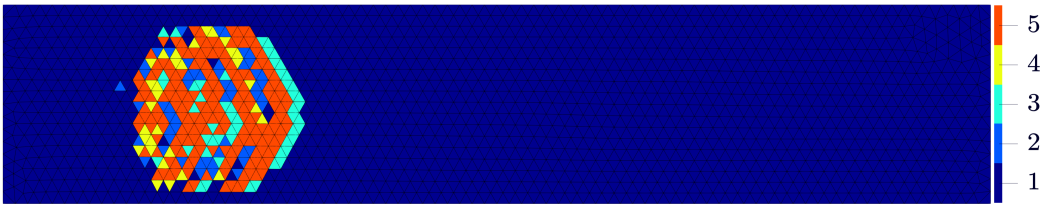
A visual comparison of the velocity snapshots with the results previously shown for a standard degree adaptivity and with the reference solution provides initial evidence of the substantial benefit of using an ANN to predict the solution at the next time and use this information in a degree adaptive process. The velocity fields at the shown times are in an excellent agreement with the reference solution and do not show the dissipation and dispersion effects that are present if the adaptivity does not account for the future time step. The degree of approximation maps in Figure 5.32 illustrate the changes introduced by using the solution at the next time step to perform the adaptivity.

The predictive approach offers several advantages by adapting the degree before the arrival of the gust perturbation.

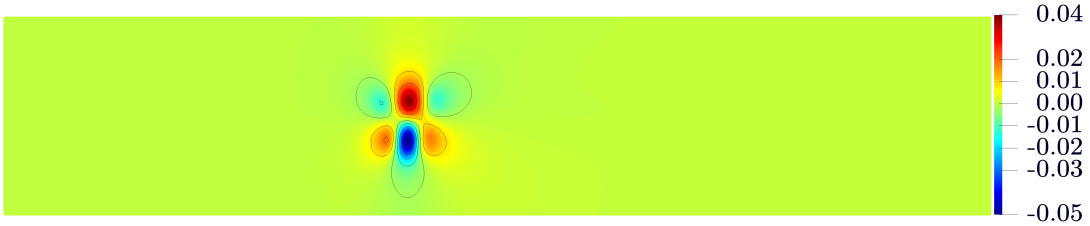
To quantify the benefits of the proposed ANN-driven degree adaptive process, Figure 5.33 show the maximum error of the velocity as a function of the non-dimensional time.

The results clearly show that around $t = 10$ the error associated with the standard adaptivity approach starts to grow and accumulates over time, reaching values higher than 40% around $t = 58$. The application of adaptivity based on the solution at time t^n only is clearly insufficient to ensure that the mesh has enough resolution where the gust will travel in the future time step. In contrast, the approach where the degree adaptivity

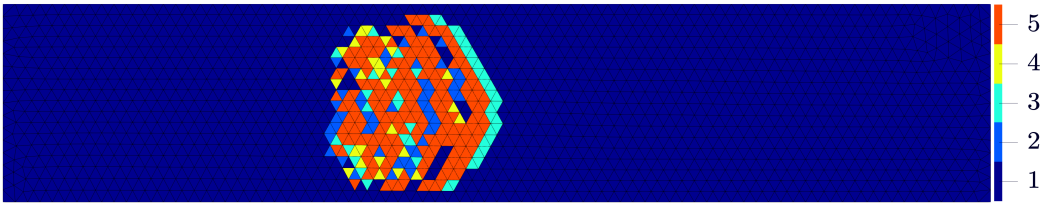
(a) Vertical velocity field (v) at $t = 16$



(b) Degree of approximation at $t = 16$

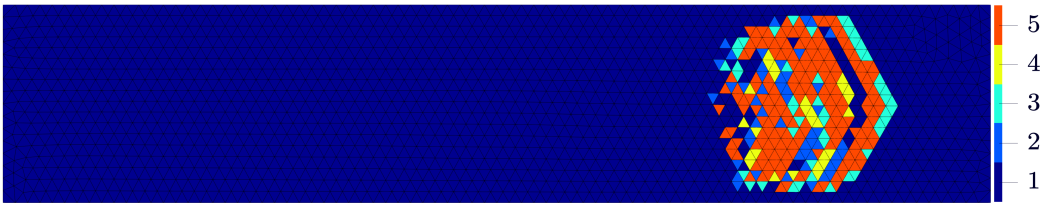


(c) Vertical velocity field (v) at $t = 32$



(d) Degree of approximation at $t = 32$

(e) Vertical velocity field (v) at $t = 64$



(f) Degree of approximation at $t = 64$

Figure 5.32: Gust in a free-stream flow: vertical velocity field (v) and degree of approximation at different instants with ANN-driven degree adaptivity.

process is driven by the trained ANN shows a significant improvement and is capable of maintaining the error below 5% during the whole simulation.

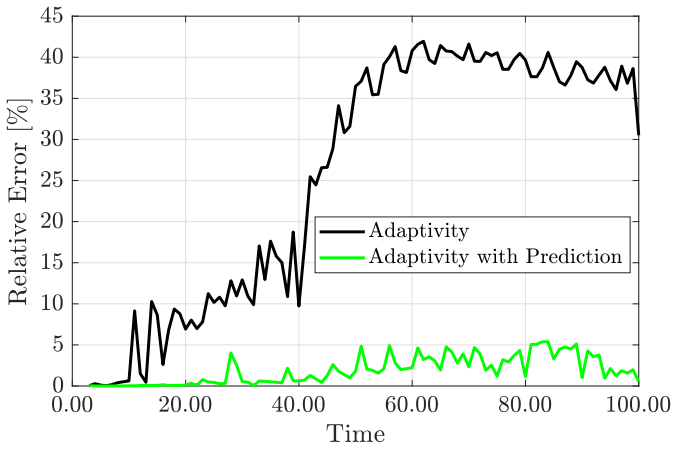


Figure 5.33: Gust in a free-stream flow: error of the velocity as a function of the non-dimensional time for the standard degree adaptivity and the ANN-driven approach.

Chapter 6

Conclusion

6.1 Summary of thesis achievements

This thesis focused on the accurate simulation of transient incompressible flows by employing the high-order HDG method, degree adaptivity, and high-order explicit first stage, singly diagonally implicit Runge-Kutta (ESDIRK) methods. The method was implemented in a Fortran 90 code and verified in two and three dimensions by performing tests to assess optimal convergence in space and time. The degree adaptive process was also tested using steady and transient problems.

In this thesis, two original contributions have been proposed.

First, a conservative projection has been proposed to enable the degree adaptive process to lower the degree of elements in time without introducing non-physical numerical artefacts. Without this projection, a standard degree adaptive process leads to non-physical oscillations in the aerodynamic quantities of interest when the degree of approximation is lowered during the time marching process. These oscillations are linked to the violation of the incompressibility condition when the degree of approximation is lowered, leading to oscillations in the pressure field. To provide further evidence about the nature of these oscillations, an adaptive process has been implemented in which

the degree of approximation is not allowed to be lowered during the time marching, leading to correct solutions. However, the extra cost of this approach makes the adaptivity not an efficient choice, especially in problems where localised transient effects travel along the domain. The proposed conservative projection completely removes the non-physical oscillations in the aerodynamic quantities of interest and enables the degree to be lowered in regions where accuracy is no longer required, leading to a more efficient use of high order approximations, only where needed. Two examples have been used to illustrate the benefits of the proposed approach and to quantify the extra accuracy and the lower computational requirements compared to a standard degree adaptive approach and an adaptive strategy where the degree is not allowed to be lowered. The first example involved the computation of the flow around two circular cylinders in tandem, whereas the second example involved a more challenging problem where a gust perturbation impinges into a NACA0012 aerofoil.

Second, a strategy to incorporate a trained ANN, that predicts the solution at the future time step, into a degree adaptive process has been proposed. The strategy enables us to predict where flow features will travel and ensure that the degree is adapted in advance. The strategy offers significant advantages when parametric analysis is to be performed, i.e. when the simulations are to be repeated by changing some inputs of the simulation. In this context, the cost of generating the data and training and tuning an ANN is justified, and the predicted capability enables efficient simulations for other combinations of the input parameters not seen during the training. The strategy for training the ANN involves using accurate data for a few simulations. A novel strategy to gather the data from a simulation and produce input and output matrices with a fixed small number of inputs and outputs is developed using a stencil of imaginary points around a mesh node. Numerical examples were performed to study the influence of the geometry and distance of the stencil on the accuracy of the predictions. Given the large amount of data gathered from a single simulation and the potential bias present in these datasets, a novel algorithm was developed to reduce the amount of data and minimise the bias that is normally associated with areas of the domain with constant

velocity and pressure fields. Numerical experiments were also presented to illustrate the effect of this process. Finally, the application to a parametric problem involving the propagation of a gust perturbation in a free-stream flow was used to illustrate and quantify the benefits of the proposed approach.

6.2 Future work

The findings detailed in this thesis have revealed multiple promising directions for future research. Some of them are:

- **Real-time learning for ANNs.** The current implementation involves training an ANN from available data and, later, deploying the trained ANN in a degree adaptive procedure. A more advanced implementation would involve launching the degree adaptive process, gather data during the simulation, train an ANN when enough data is gathered and deploy the trained ANN in real time. During the first stages of the simulation it is anticipated that multiple adaptivity loops would be required to ensure that flow features are not lost but at some point, once the ANN is deployed, this repetition would no longer be needed, as demonstrated in this work. This more advanced strategy would require monitoring the inputs to ensure that they lie within the limits of the data used to train the ANN and, when they are outside the ANN could be retrained using data continuously gathered from the current simulation. It is anticipated that retraining the ANN would not require a major cost because the weights of a previously trained ANN could be use to initialise the training.

- **Cross-scenario prediction:** The examples considered in this work involve parametric problems and the training data and the predictions are performed using the same parameters. Investigating the ability of ANN trained on specific scenarios (e.g., particular geometries or flow conditions) to predict outcomes in different scenarios would be beneficial. As an example, it would be worth investigating

the accuracy of the predictions of an ANN trained using data gathered from a simulation of the flow around a circular cylinder to predict the flow around an aerofoil. If the required accuracy is not obtained directly, transfer learning techniques could be used to accelerate the training of an ANN for new problem configurations, potentially reducing the computational cost of simulating new scenarios.

- **Extension to three-dimensions.** The HDG solver implemented has been applied and tested for two and three dimensional problems. However, the numerical examples that involve training an ANN have been restricted to two dimensions to avoid performing expensive three dimensional simulations to gather data for the training. Given the benefits shown in the examples, it would be worth investigating the extension to three dimensions, which would only require adapting the stencil to three dimensions. In this context, it would be interesting to assess the increase in computational time required to train the ANN and also the architecture of the ANN that is capable of providing the required accuracy in three dimensions.

- **Applications.** This work has focused on the solution of transient incompressible laminar viscous flows. The strategy developed is general and could be applied to other flow problems such as transient compressible viscous flow and turbulent compressible and incompressible flow. The extension to compressible flows would require a careful choice of the outputs of the ANN to ensure that an error indicator can be built using the super-convergent postprocess of the solution. In addition, for large Reynolds number it is anticipated that the super-convergent properties might be lost and other a-posteriori error indicators might be needed.

  Extending the adaptive HDG-ANN framework to incorporate additional physical phenomena would greatly enhance its applicability to real-world engineering challenges. This could include:

  - Coupling with convection-diffusion equations to simulate heat transfer or

species transport in fluids.

- Integration with structural mechanics for fluid-structure interaction problems.
- Incorporation of acoustic models for aeroacoustic simulations.

These extensions would require careful consideration of the interplay between fluid dynamics and the additional physical processes, as well as the development of efficient numerical strategies to handle the increased complexity of the coupled systems.

Bibliography

Ainsworth, M., Monk, P., and Muniz, W. (2006). Dispersive and dissipative properties of discontinuous Galerkin finite element methods for the second-order wave equation. Journal of Scientific Computing, 27:5–40.

Alamri, Y. and Ketcheson, D. (2024). Very high-order A-stable stiffly accurate diagonally implicit Runge-Kutta methods with error estimators. Journal of Scientific Computing, 100.

Alauzet, F., Frey, P. J., George, P.-L., and Mohammadi, B. (2007). Transient fixed point-based unstructured mesh adaptation. International Journal for Numerical Methods in Fluids, 54(6-8):789–810.

Ali, Z., Dhanasekaran, P. C., Tucker, P. G., Watson, R., and Shahpar, S. (2017). Optimal multi-block mesh generation for CFD. International Journal of Computational Fluid Dynamics, 31(4-5):195–213.

Amestoy, P. R., Duff, I. S., L'Excellent, J.-Y., and Koster, J. (2001). MUMPS: a general purpose distributed memory sparse solver. International Conference on Applied Parallel Computing, pages 121–130.

Arnold, D. N. (1982). An interior penalty finite element method with discontinuous elements. SIAM journal on numerical analysis, 19(4):742–760.

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., et al. (2019). PETSc users manual. Argonne National Lab.(ANL), Argonne, IL (United States).

Bassi, F., Crivellini, A., Di Pietro, D. A., and Rebay, S. (2007). An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows. Computers & Fluids, 36(10):1529–1546.

Baumann, C. E. and Oden, J. T. (1999). A discontinuous hp finite element method for convection—diffusion problems. Computer Methods in Applied Mechanics and Engineering, 175(3-4):311–341.

Biedron, R. T., Carlson, J.-R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., et al. (2016). FUN3D Manual: 12.9. Technical Report TM-2016-219012, NASA.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2020). Machine learning for fluid mechanics. Annual Review of Fluid Mechanics, 52:477–508.

Campagne, G., Hassan, O., Morgan, K., and Sørensen, K. (2010). Higher–order aerodynamic computations using an edge based finite volume scheme. In ADIGMA-A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications, pages 309–325. Springer.

Cesmelioglu, A., Cockburn, B., and Qiu, W. (2017). Analysis of a hybridizable discontinuous Galerkin method for the steady-state incompressible Navier-Stokes equations. Mathematics of Computation, 86(306):1643–1670.

Chalot, F., Dagrau, F., Mallet, M., Normand, P., and Yser, P. (2015). Higher-order RANS and DES in an industrial stabilized finite element code. IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach: Results of a Collaborative Research Project Funded by the European Union, 2010-2014, pages 489–519.

Chalot, F. and Normand, P.-E. (2010). Higher-order stabilized finite elements in an industrial Navier-Stokes code. In Kroll, N., Bieler, H., Deconinck, H., Couaillier, V., van der Ven, H., and Sørensen, K., editors, ADIGMA - A European Initiative

on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications, pages 145–165, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cockburn, B. (2017). Discontinuous Galerkin methods for computational fluid dynamics. Encyclopedia of Computational Mechanics Second Edition, pages 1–63.

Cockburn, B. and Fu, G. (2017). Devising superconvergent HDG methods with symmetric approximate stresses for linear elasticity by M-decompositions. IMA Journal of Numerical Analysis, 38(2):566–604.

Cockburn, B. and Gopalakrishnan, J. (2005). Incompressible finite elements via hybridization. I. The Stokes system in two space dimensions. SIAM Journal on Numerical Analysis, 43(4):1627–1650.

Cockburn, B. and Gopalakrishnan, J. (2009). The derivation of hybridizable discontinuous Galerkin methods for Stokes flow. SIAM Journal on Numerical Analysis, 47(2):1092–1125.

Cockburn, B., Gopalakrishnan, J., and Lazarov, R. (2009a). Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems. SIAM Journal on Numerical Analysis, 47(2):1319–1365.

Cockburn, B., Gopalakrishnan, J., Nguyen, N. C., Peraire, J., and Sayas, F.-J. (2011). Analysis of HDG methods for Stokes flow. Mathematics of Computation, 80(274):723–760.

Cockburn, B., Hou, S., and Shu, C.-W. (1990). The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. The multidimensional case. Mathematics of Computation, 54(190):545–581.

Cockburn, B., Kanschat, G., and Schötzau, D. (2005). A locally conservative LDG method for the incompressible Navier-Stokes equations. Mathematics of computation, 74(251):1067–1095.

Cockburn, B., Kanschat, G., and Schötzau, D. (2009b). An equal-order dg method for the incompressible navier-stokes equations. Journal of Scientific Computing, 40(1):188–210.

Cockburn, B., Lin, S.-Y., and Shu, C.-W. (1989). TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. Journal of computational Physics, 84(1):90–113.

Cockburn, B., Nguyen, N. C., and Peraire, J. (2010). A comparison of HDG methods for Stokes flow. Journal of Scientific Computing, 45(1-3):215–237.

Cockburn, B. and Shi, K. (2013). Conditions for superconvergence of HDG methods for Stokes flow. Mathematics of Computation, 82(282):651–671.

Cockburn, B. and Shi, K. (2014). Devising HDG methods for Stokes flow: An overview. Computers & Fluids, 98:221–229.

Cockburn, B. and Shu, C. (1991). The Runge—Kutta local projection P^1-discontinuous Galerkin method for scalar conservation laws. Rairo Math. Model. Numer. Anal. Model. Math. et Anal. Numer, 25.

Cockburn, B. and Shu, C.-W. (1989). TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. Mathematics of computation, 52(186):411–435.

Cockburn, B. and Shu, C.-W. (1998a). The local discontinuous Galerkin method for time-dependent convection-diffusion systems. SIAM journal on numerical analysis, 35(6):2440–2463.

Cockburn, B. and Shu, C.-W. (1998b). The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. Journal of computational physics, 141(2):199–224.

Curtiss, C. F. and Hirschfelder, J. O. (1952). Integration of stiff equations. Proceedings of the National Academy of Sciences of the United States of America, 38(3):235.

Deng, G., Pique, J., Queutey, P., and Visonneau, M. (1996). Navier-Stokes equations for incompressible flows: Finite-difference and finite-volume methods. In Handbook of Computational Fluid Mechanics, pages 25–97. Elsevier.

Donea, J. and Huerta, A. (2003). Finite element methods for flow problems. John Wiley & Sons.

Dzanic, T., Mittal, K., Kim, D., Yang, J., Petrides, S., Keith, B., and Anderson, R. (2024). DynAMO: Multi-agent reinforcement learning for dynamic anticipatory mesh optimization with applications to hyperbolic conservation laws. Journal of Computational Physics, 506:112924.

Díez, P. and Huerta, A. (1999). A unified approach to remeshing strategies for finite element h-adaptivity. Computer Methods in Applied Mechanics and Engineering, 176(1):215–229.

Ekaterinaris, J. A. (2005). High-order accurate, low numerical diffusion methods for aerodynamics. Progress in Aerospace Sciences, 41(3-4):192–300.

Ekelschot, D., Moxey, D., Sherwin, S., and Peiró, J. (2017). A p-adaptation method for compressible flow problems using a goal-based error indicator. Computers & Structures, 181:55–69.

Ethier, C. R. and Steinman, D. A. (1994). Exact fully 3D Navier–Stokes solutions for benchmarking. International Journal for Numerical Methods in Fluids, 19(5):369–375.

Ferrer, E. and Willden, R. (2011). A high order discontinuous Galerkin finite element solver for the incompressible Navier–Stokes equations. Computers & Fluids, 46(1):224–230.

Fidkowski, K. J. and Darmofal, D. L. (2007). A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier–Stokes equations. Journal of Computational Physics, 225(2):1653–1672.

Franca, L. P. and Frey, S. L. (1992). Stabilized finite element methods: II. The incompressible Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering, 99(2-3):209–233.

Gerhold, T. (2005). Overview of the hybrid RANS code TAU. In MEGAFLOW-Numerical Flow Simulation for Aircraft Design, pages 81–92. Springer.

Giacomini, M., Karkoulias, A., Sevilla, R., and Huerta, A. (2018). A superconvergent HDG method for Stokes flow with strongly enforced symmetry of the stress tensor. Journal of Scientific Computing, 77(3):1679–1702.

Giacomini, M., Sevilla, R., and Huerta, A. (2020). Tutorial on hybridizable discontinuous Galerkin (HDG) formulation for incompressible flow problems, pages 163–201. Springer International Publishing, Cham.

Giorgiani, G., Fernández-Méndez, S., and Huerta, A. (2013). Hybridizable discontinuous Galerkin p-adaptivity for wave propagation problems. International Journal for Numerical Methods in Fluids, 72(12):1244–1262.

Giorgiani, G., Fernández-Méndez, S., and Huerta, A. (2014). Hybridizable discontinuous Galerkin with degree adaptivity for the incompressible Navier–Stokes equations. Computers & Fluids, 98:196–208.

Golubev, V., Dreyer, B., Hollenshade, T., and Visbal, M. (2009). High-accuracy viscous simulation of gust-airfoil nonlinear aeroelastic interaction. In 39th AIAA fluid dynamics conference, page 4200.

Gross, R., Chalot, F., Courty, J.-C., Mallet, M., Tran, D., Arnal, D., and Vermeersch, O. (2015). Automatic transition prediction in an industrial Navier-Stokes solver using higher-order finite elements. In 45th AIAA Fluid Dynamics Conference, page 2621.

Gürkan, C., Kronbichler, M., and Fernández-Méndez, S. (2019). eXtended hybridizable discontinuous Galerkin for incompressible flow problems with unfitted meshes and

interfaces. International Journal for Numerical Methods in Engineering, 117(7):756–777.

Guyan, R. (1965). Reduction of stiffness and mass matrices. AIAA Journal, 3(2):380–380.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik, 2(1):84–90.

Haykin, S. (2009). Neural Networks and Learning Machines. Pearson, New York, 3 edition.

Huergo, D., de Frutos, M., Jané, E., Marino, O. A., Rubio, G., and Ferrer, E. (2024). Reinforcement learning for anisotropic p-adaptation and error estimation in high-order solvers. arXiv preprint arXiv:2407.19000.

Hundsdorfer, W. and Verwer, J. G. (2013). Numerical solution of time-dependent advection-diffusion-reaction equations, volume 33. Springer Science & Business Media.

Jørgensen, J. B., Kristensen, M. R., and Thomsen, P. G. (2018). A family of ESDIRK integration methods. arXiv preprint arXiv:1803.01613.

Kennedy, C. and Carpenter, M. (2016). Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review. Technical Report TM-2016-219173, NASA.

Kennedy, C. A. and Carpenter, M. H. (2003). Additive Runge–Kutta schemes for convection–diffusion–reaction equations. Applied Numerical Mathematics, 44(1-2):139–181.

Kirby, R. M., Sherwin, S. J., and Cockburn, B. (2012). To CG or to HDG: a comparative study. Journal of Scientific Computing, 51:183–212.

Kocis, L. and Whiten, W. J. (1997). Computational investigations of low-discrepancy sequences. ACM Transactions on Mathematical Software (TOMS), 23(2):266–294.

Komala-Sheshachala, S., Sevilla, R., and Hassan, O. (2020). A coupled HDG-FV scheme for the simulation of transient inviscid compressible flows. Computers & Fluids, 202:104495.

Kompenhans, M., Rubio, G., Ferrer, E., and Valero, E. (2016). Comparisons of p-adaptation strategies based on truncation-and discretisation-errors for high order discontinuous Galerkin methods. Computers & Fluids, 139:36–46.

Kovasznay, L. I. G. (1948). Laminar flow behind a two-dimensional grid. Mathematical Proceedings of the Cambridge Philosophical Society, 44(1):58–62.

Kværnø, A. (2004). Singly diagonally implicit Runge–Kutta methods with an explicit first stage. BIT Numerical Mathematics, 44:489–502.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553):436–444.

Lehrenfeld, C. and Schöberl, J. (2016). High order exactly divergence-free hybrid discontinuous Galerkin methods for unsteady incompressible flows. Computer Methods in Applied Mechanics and Engineering, 307:339–361.

Leng, H. (2021). Adaptive HDG methods for the steady-state incompressible Navier–Stokes equations. Journal of Scientific Computing, 87(1):37.

Li, J., Lin, X., and Chen, Z. (2022). Finite volume methods for the incompressible Navier-stokes equations, volume 2022. Springer.

Liu, J.-G. and Shu, C.-W. (2000). A high-order discontinuous Galerkin method for 2D incompressible flows. Journal of Computational Physics, 160(2):577–596.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 6(4):525–533.

Montlaur, A., Fernandez-Mendez, S., and Huerta, A. (2008). Discontinuous Galerkin methods for the Stokes equations using divergence-free approximations. International Journal for Numerical Methods in Fluids, 57(9):1071–1092.

Montlaur, A., Fernandez-Mendez, S., Peraire, J., and Huerta, A. (2010). Discontinuous Galerkin methods for the Navier–Stokes equations using solenoidal approximations. International journal for numerical methods in fluids, 64(5):549–564.

Morgan, K., Peraire, J., Peiro, J., and Hassan, O. (1991). The computation of three-dimensional flows using unstructured grids. Computer Methods in Applied Mechanics and Engineering, 87(2-3):335–352.

Nguyen, N. C., Peraire, J., and Cockburn, B. (2010). A hybridizable discontinuous Galerkin method for Stokes flow. Computer Methods in Applied Mechanics and Engineering, 199(9-12):582–597.

Nguyen, N. C., Peraire, J., and Cockburn, B. (2011). An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations. Journal of Computational Physics, 230(4):1147–1170.

Nogueira, X., Colominas, I., Cueto-Felgueroso, L., and Khelladi, S. (2010). On the simulation of wave propagation with a higher-order finite volume scheme based on reproducing kernel methods. Computer Methods in Applied Mechanics and Engineering, 199(23-24):1471–1490.

Paipuri, M., Fernández-Méndez, S., and Tiago, C. (2018). Comparison of high-order continuous and hybridizable discontinuous Galerkin methods for incompressible fluid flow problems. Mathematics and computers in simulation, 153:35–58.

Patankar, S. V. and Spalding, D. B. (1983). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In Numerical prediction of flow, heat transfer, turbulence and combustion, pages 54–73. Elsevier.

Peraire, J. and Persson, P.-O. (2008). The compact discontinuous Galerkin (CDG) method for elliptic problems. SIAM Journal on Scientific Computing, 30(4):1806–1824.

Quartapelle, L. (2013). Numerical solution of the incompressible Navier-Stokes equations, volume 113. Birkhäuser.

Quarteroni, A. (2017). Numerical models for differential problems. Springer.

Rhebergen, S. and Wells, G. N. (2018). A hybridizable discontinuous Galerkin method for the Navier–Stokes equations with pointwise divergence-free velocity field. Journal of Scientific Computing, 76(3):1484–1501.

Sanz-Serna, J. M., Verwer, J. G., and Hundsdorfer, W. (1986). Convergence and order reduction of Runge-Kutta schemes applied to evolutionary problems in partial differential equations. Numerische Mathematik, 50:405–418.

Schenk, O. and Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with PARDISO. Future Generation Computer Systems, 20(3):475–487.

Sevilla, R., Giacomini, M., Karkoulias, A., and Huerta, A. (2018). A superconvergent hybridisable discontinuous Galerkin method for linear elasticity. International Journal for Numerical Methods in Engineering, 116(2):91–116.

Sevilla, R., Hassan, O., and Morgan, K. (2013). An analysis of the performance of a high-order stabilised finite element method for simulating compressible flows. Computer Methods in Applied Mechanics and Engineering, 253:15–27.

Sevilla, R. and Huerta, A. (2016). Tutorial on Hybridizable Discontinuous Galerkin (HDG) for Second-Order Elliptic Problems, pages 105–129. Springer International Publishing, Cham.

Sevilla, R. and Huerta, A. (2018). HDG-NEFEM with degree adaptivity for Stokes flows. Journal of Scientific Computing, 77(3):1953–1980.

Shields, M. D. and Zhang, J. (2016). The generalization of Latin hypercube sampling. Reliability Engineering & System Safety, 148:96–108.

Soon, S.-C., Cockburn, B., and Stolarski, H. K. (2009). A hybridizable discontinuous Galerkin method for linear elasticity. International journal for numerical methods in engineering, 80(8):1058–1092.

Temam, R. (2001). Navier-Stokes equations: theory and numerical analysis. American Mathematical Soc.

Wang, C. Y. (1991). Exact solutions of the steady-state Navier-Stokes equations. Annual Review of Fluid Mechanics, 23(1):159–177.

Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., et al. (2013). High-order CFD methods: current status and perspective. International Journal for Numerical Methods in Fluids, 72(8):811–845.

Whiting, C. H. and Jansen, K. E. (2001). A stabilized finite element method for the incompressible Navier–Stokes equations using a hierarchical basis. International Journal for Numerical Methods in Fluids, 35(1):93–116.

Xie, Z. Q., Sevilla, R., Hassan, O., and Morgan, K. (2013). The generation of arbitrary order curved meshes for 3D finite element analysis. Computational Mechanics, 51:361–374.

Yakovlev, S., Moxey, D., Kirby, R. M., and Sherwin, S. J. (2015). To CG or to HDG: A comparative study in 3D. Journal of Scientific Computing, pages 1–29.