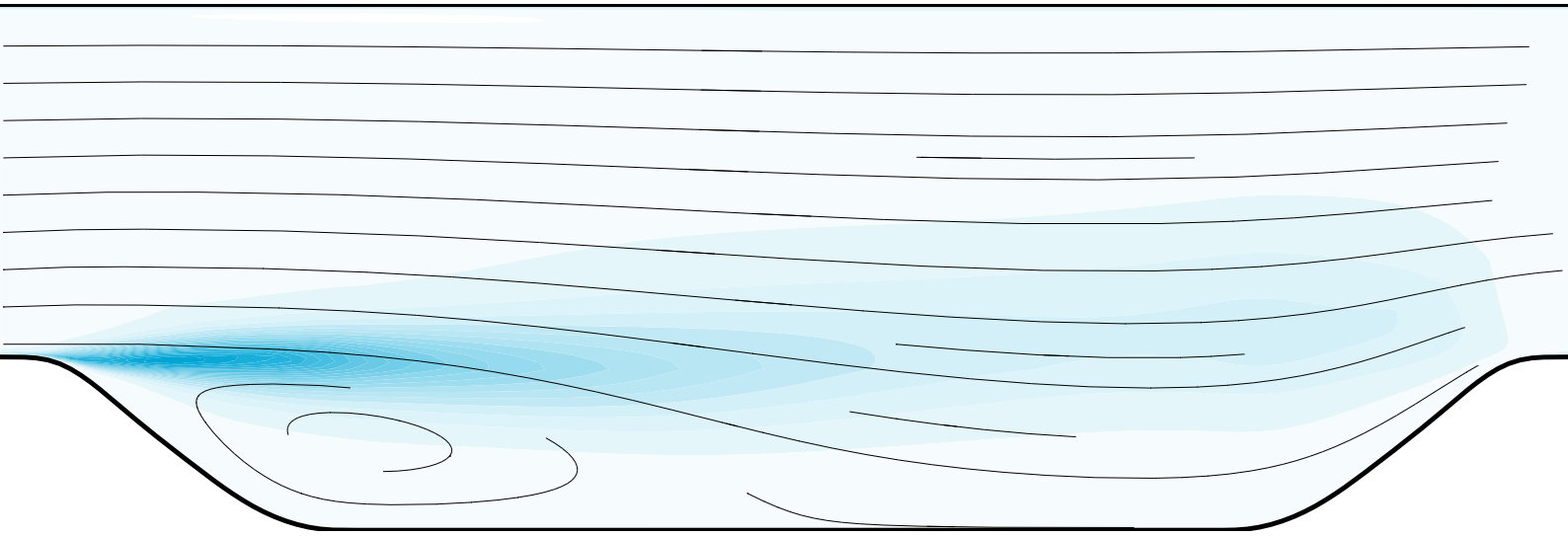
Master of Science Thesis



Field inversion and machine learning in turbulence modeling

A. F. van Korlaar

February 22, 2019

Aerodynamics

TU Delft

Faculty of Aerospace Engineering

Delft University of Technology

Field inversion and machine learning in turbulence modeling

by

A. F. van Korlaar

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday February 22, 2019 at 10:00 AM.

Student number:	4295951	
Project duration:	May, 2018 – February, 2019	
Supervisor	Dr. R.P. Dwight	TU Delft
Thesis committee:	Prof. Dr. S. Hickel	TU Delft
	Dr.ir. M. Pini	TU Delft
	Dr. S.J. Hulshoff	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Turbulence closure models will continue to be necessary in order to perform computationally affordable simulations in the foreseeable future. It is expected that Reynolds-averaged Navier-Stokes (RANS) turbulence models will still be useful with the further development of the more accurate, but computationally expensive large eddy simulation (LES), especially in industry. The use of the robust but often inaccurate linear eddy viscosity closures is still widespread in industry. More complex closure models, such as Reynolds stress models and nonlinear eddy viscosity models, provide a more general description of the underlying physics of turbulent flows. Nevertheless, because of implementational difficulties or failure to provide consistent improvements over the more robust linear models, RANS turbulence modeling is considered to have reached a plateau. In the past few years, the availability of high-fidelity datasets, the increased accuracy of machine learning algorithms, and the rise in computational power led to the proposal of several data-driven approaches to turbulence modeling. The general idea is to use experimental and high-fidelity data to develop or enhance RANS turbulence models, instead of employing an approach purely based on physics. As in any emerging field, there are many possibilities for further developing the novel approaches to data-driven turbulence modeling. Recent work combined machine learning with statistical inversion. First, a spatially varying correction is applied to the RANS model and optimized by minimizing the discrepancy between the RANS output and the data for several flows. Machine learning is used to approximate a function between a set of flow features and the inferred corrections. The aim of this work is to further investigate this methodology, called the paradigm of field inversion and machine learning, in a broader set of test cases by inferring a spatially varying correction to the production term of the ω-equation in the $k-\omega$ model and to the eigenvalues of the Reynolds stress tensor. The gradients of this high-dimensional optimization process are obtained by implementing the continuous adjoint of the $k-\omega$ model in OpenFOAM. Gaussian processes and random forests are then used to approximate a function between mean flow features and the inferred corrections. It was found that both formulations are able to accurately infer the mean velocities and related quantities of interest, but that the inferred corrective terms are often non-unique or not physically interpretable. For several flow cases, the corrective terms were able to generalize to unseen Reynolds number and flow geometries.

Preface

I would like to express my gratitude to my supervisor, Dr. Richard Dwight. First of all, for offering me the opportunity to work on the subject I am passionate about and for introducing me to the beautiful world of adjoints, but also for being an exceptionally involved and enthusiastic supervisor. Your quick replies and willingness to meet frequently have been more than helpful for getting this project finished. I enjoyed our frequent conversations, which often went back to the fundamentals, and I believe they had a positive effect on shaping my approach to research. Secondly, I would like to thank Martin Schmelzer for getting me started with OpenFOAM and helping me with the other practicalities necessary to get going. I would also like to thank Wouter Edeling, for the discussions we had in the early phases of the project. My gratitude also goes to my friends, for the good conversations, the laughter, and for getting my mind of this project from time to time. A special mention should go to Joël for the five years of companionship. Who would have thought back then that we would finish the same studies at the same time. Thanks also to the basement residents, for the pleasant lunches and tea breaks, and for bringing some warmth into the space. Last but not least, I would like to thank my parents, Adriaan and Jantien, and my sister Karien, for their continuous and unconditional love and support.

A. F. van Korlaar
Delft, February 2019

Contents

List of Figures

List of Tables

Acronyms

ARSM algebraic Reynolds stress model

CFD computational fluid dynamics

CG conjugate gradients

DNS direct numerical simulation

EARSM explicit algebraic Reynolds stress model

EnKF ensemble Kalman filter

FIML field inversion and machine learning

k-NN k-nearest neighbours

LES large eddy simulation

LEVM linear eddy viscosity model

LOOCV leave-one-out cross-validation

MAP maximum a posteriori

MCMC Markov chain Monte Carlo

NLEVM non-linear eddy viscosity model

PDE partial differential equation

PIV particle image velocimetry

RANS Reynolds-averaged Navier-Stokes

RBF radial basis function

RSM Reynolds stress model

SVM support vector machine

VLES very large eddy simulation

Nomenclature

Latin symbols

$\mathcal{P}_{ij}$	Production tensor
$\mathcal{R}_{ij}$	Pressure-rate-of-strain tensor
$\mathcal{T}_{ijk}$	Reynolds stress flux
Re	Reynolds number
Re_τ	Reynolds number based on friction velocity
a_{ij}	Anisotropic Reynolds stresses
b_{ij}	Normalized Reynolds stress anisotropy
C_β	Prior covariance matrix
C_m	Observational covariance matrix
C_{α_c}	α-th barycentric coordinate
J	Objective function
k	Turbulent kinetic energy
k_a	Adjoint turbulent kinetic energy
L	Characteristic lengthscale
P	Instantaneous pressure
p	Mean pressure
q	Adjoint mean pressure
q	Unit quaternion
R_{ij}	Reynolds stress tensor
S_{ij}	Strain rate tensor
t	Time
U	Characteristic velocity
u'_i	Adjoint mean velocity
V_i	Instantaneous velocity
v_i	Mean velocity
v'_i	Fluctuating velocity
x	Cartesian coordinate

Greek symbols

β	Corrective term
β_P	Production term correction
$\beta_{C_{\alpha c}}$	Eigenvalue correction, α-th barycentric coordinate
δ_{ij}	Kronecker delta
η	Kolmogorov lengthscale
η	Step size
ν	Kinematic viscosity
ω	Specific turbulent dissipation rate
ω_a	Adjoint specific turbulent dissipation rate
Ω_{ij}	Rotation rate tensor
σ_β	Prior variance

σ_m	Observational variance
τ_η	Kolmogorov timescale
ε	Rate of dissipation of turbulent kinetic energy
ε_{ij}	Dissipation tensor
u_η	Kolmogorov velocity

1 Introduction

1.1. Background

Simulation of turbulent flows is an important topic of research, as it plays an important role in a broad range of applications in science and engineering. The equations governing the behavior of turbulence, the Navier-Stokes equations, can be solved without modeling the turbulence in direct numerical simulations (DNS). However, this is excessively expensive in many cases of interest in science and engineering [116]. Therefore, turbulence closure models will continue to be necessary in order to perform computationally affordable simulations in the foreseeable future [139]. In large eddy simulations (LES) the smallest scales of turbulence are modeled, whereas the larger scales are solved for. Solving instead for the Reynolds averaged Navier-Stokes equations (RANS) is still the most widely used approach in industry, mainly because of its relatively low computational cost. It is expected that RANS turbulence models will still be useful with the further development of the more accurate, but computationally expensive LES, especially in industry [47].

The most commonly used RANS models use the Boussinesq assumption, in which a linear relation between the Reynolds stress and the mean velocity gradients is assumed. These models perform well in certain classes of flows, but fail to model flows which are often present in cases of engineering interest, such as curved surfaces, rotating flows, and secondary flows in corners. The more general Reynolds stress models (RSMs) or non-linear eddy viscosity models (NLEVMs) have not seen widespread use in industry because of difficulties in their implementation or failure to provide a consistent improvement over linear eddy viscosity models. This is one of the reasons why RANS turbulence modeling is considered to have reached a plateau [139].

Measurement techniques such as particle image velocimetry (PIV) and the use of DNS in academic settings have provided high-fidelity datasets over the past decades. These datasets, combined with the increased accuracy of machine learning algorithms and the rise in computational power, created an opportunity for data-driven approaches to turbulence modeling. The general idea is to use experimental data and the results from high-fidelity simulations to help develop or enhance RANS turbulence models, instead of employing an approach purely based on physical reasoning. As in any emerging field, there are many possibilities for further developing the novel approaches to data-driven turbulence modeling. For example, Weatheritt and Sandberg [159, 160] use symbolic regression and gene expression programming to obtain analytical expressions for the anisotropy tensor. Ling et al. [82] and Wang et al. [157] use a different method, and directly predict the Reynolds stress tensor or its discrepancy from the data. Another approach, proposed by Duraisamy et al. [29], Parish and Duraisamy [107], combines machine learning with statistical field inversion. In this method, called the paradigm of field inversion and machine learning, a spatially varying correction is applied to a physical model. Using an optimization process, the optimal correction is inferred by minimizing discrepancy between the model output and a high-fidelity dataset. After this field inversion phase is performed on a number of cases, a machine learning algorithm is used to approximate a function between a set of mean flow features and the corrective term. This paradigm will be the main focus of this work.

The paradigm of field inversion and machine learning has been developed and examined in a variety of works [29, 34, 52, 106, 107, 152]. In most of these works, the spatially varying corrective term is applied to the production term in the Spalart-Allmaras model, or the k- or ω-equation in the $k-\omega$ model. A disadvantage of this approach is that it does not cover the complete model form error introduced by the Boussinesq assumption, but only corrects the eddy viscosity. There have been some investigations in inferring corrections to the

eigenvalues. However, these were limited to simple flows cases such as fully-developed turbulent channel flow. All these works used the discrete adjoint for the field inversion phase. Furthermore, most of the investigations in using the inferred corrective terms in a predictive setting have been performed only for different Reynolds numbers or similar geometries. The novelty of this work is twofold: firstly, the continuous adjoint of the $k-\omega$ model is used for the field inversion phase in the paradigm of field inversion and machine learning, using both corrections to the production term in the ω-equation and the eigenvalues of the Reynolds stress tensor. Secondly, it investigates the inferred corrective terms and corresponding quantities of interest for a variety of canonical flows and studies the possibility of training random forests and Gaussian processes to predict the corrective term on different flow cases.

1.2. Research objectives

Two research questions are divided into six sub-questions, and are formulated as follows.

- Q1 Can the paradigm of field inversion and machine learning in its current form, as proposed by Parish and Duraisamy [107] and Singh et al. [138], be extended using the continuous adjoint of the $k-\omega$ model for the field inversion phase?
  - SQ1 How does the one-shot optimization compare to the complete approach in terms of convergence and accuracy?
  - SQ2 How does the paradigm perform in terms of quantifying the uncertainty in the inferred corrective functions and the resulting quantities of interest in a computationally tractable manner?
- Q2 What are the strengths and limitations of formulating the corrective term as corrections to the production term in the ω-equation and the eigenvalues of the Reynolds stress tensor?
  - SQ3 Is it possible to accurately infer the mean velocities and other quantities of interest using the given corrective term formulations?
  - SQ4 How do the two formulations of the corrective term compare in terms of physical interpretability?
  - SQ5 Can the inferred corrective terms generalize to other Reynolds numbers and flow geometries using machine learning algorithms?
  - SQ6 Does including flow cases with similar flow features in the training set as are present in the test case lead to a notable improvement in the accuracy of the machine learning prediction?

The research objective is

"*To develop a new data-driven turbulence model based on the paradigm of field inversion and machine learning using the continuous adjoint of the $k-\omega$ model and to investigate the formulation of the corrective function in terms of their ability to infer the quantities of interest, interpretability, and generalizability.*"

These research questions and the extent to which the research objective is accomplished will be discussed in the conclusion.

1.3. Thesis outline

Chapter 2 gives a broad overview of the field of RANS turbulence modeling and the recent advances in data-driven techniques. The chapter is concluded by a brief outline of the paradigm considered in this thesis. The theoretical basis for the field inversion phase of the paradigm is given in chapter 3. Furthermore, a simple model problem is introduced, which is used throughout the theoretical chapters to illustrate the most important steps. Chapter 4 provides a theoretical background for the machine learning phase of the paradigm. The flow cases used to train and test the framework are described in chapter 5. The theory of the first chapters is used in chapter 6 to justify the choices made in this work. Furthermore, the expressions resulting from the derivations and the details of the implementations are given. The results are divided into two chapters, chapter 7 giving the results of the field inversion and chapter 8 those of the machine learning phase. Finally, conclusions and recommendations for future work are described in chapter 9.

2

Turbulence modeling

This chapter gives a broad overview of the field of turbulence modeling, with the main focus on RANS turbulence models. Section 2.1 gives a general introduction to the problems faced in simulating turbulent flows. Common approaches to modeling turbulence and their strengths and shortcomings are described in section 2.2. Section 2.3 describes the properties and representation of the anisotropy tensor, which will be used later in the formulation of the corrective terms. Recent advancements in data-driven approaches to RANS turbulence modeling are discussed in section 2.4, and the choice for the method considered in this thesis is justified. The chapter is concluded by an introduction to the paradigm considered in this thesis in section 2.5, before discussing the details in the coming chapters.

2.1. Problem formulation

Turbulent flows are characterized by a chaotic, unsteady nature over a wide range of length- and time-scales. An important characteristic of turbulence is its ability to mix fluids better than laminar flows. Turbulent flows are governed by the Navier-Stokes equations, which in incompressible form and without body forces are given by

$$\frac{\partial V_i}{\partial t} + V_j \frac{\partial V_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 V_i}{\partial x_j^2}, \tag{2.1}$$

where V_i is the i-th component of the instantaneous velocity and P is the instantaneous pressure. In principle, the equations can be discretized and solved without modeling the turbulence. The Reynolds number is a non-dimensional number defined by the ratio between the inertial forces and the viscous forces:

$$\mathrm{Re} = \frac{UL}{\nu}, \tag{2.2}$$

where U and L are the characteristic velocity and length, respectively, and ν is the kinematic viscosity. Kolmogorov's hypotheses and the energy cascade state that kinetic energy enters a turbulent flow at the largest scales of motion. The large scales then break down into smaller scales through inviscid processes until the energy is dissipated at the smallest scales. The Kolmogorov scales, which characterize the smallest length-, velocity-, and time-scales, can be derived using dimensional arguments and are given by

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}}, \qquad u_\eta = (\varepsilon\nu)^{\frac{1}{4}}, \qquad \text{and} \qquad \tau_\eta = \left(\frac{\nu}{\varepsilon}\right)^{\frac{1}{2}}, \tag{2.3}$$

respectively, where ε is the dissipation rate. Assuming that the dissipation rate ε is equal to the rate of production of the kinetic energy, the dissipation rate can be related to the characteristic lengthscales of the largest scales of motion,

$$\varepsilon \sim \frac{U^3}{L}. \tag{2.4}$$

The ratios between the smallest and the largest lengthscales can then be derived to be

$$\frac{\eta}{L} \sim \mathrm{Re}^{-\frac{3}{4}}, \qquad \frac{u_\eta}{U} \sim \mathrm{Re}^{-\frac{1}{4}}, \qquad \frac{\tau_\eta}{T} \sim \mathrm{Re}^{-\frac{1}{2}}, \tag{2.5}$$

and thus scale with the Reynolds number. A computational mesh is required which both resolves the smallest scales of motion and is large enough for the largest scales. Furthermore, the timestep should be small enough to resolve the smallest timescales. Therefore, even with the increase in computational power, direct numerical simulation of turbulent flows is infeasible for industrial applications, where the Reynolds number can be of order 10^8.

2.2. Turbulence modeling

2.2.1. Large eddy simulation

In large eddy simulation (LES), a spatial filtering operation is applied to the velocity field. The resolved scales of turbulence are simulated directly, whereas it is assumed that the subgrid-scales have a more universal nature. A closure model is thus needed to model the effect of the small scales on the resolved larger scales. The wavenumber at which this distinction is made is called the cut-off wavenumber. Usually, around 80% of the kinetic energy is resolved [116, p. 560]. In very large eddy simulation (VLES), a large part of the energy is is in the subgrid-scales, which allows for the use of coarser grids and thus a lower computational cost, often at the expense of accuracy.

2.2.2. Reynolds-averaged Navier Stokes

Another option is to instead solve for the mean flow quantities. The Reynolds-averaged Navier-Stokes (RANS) equations are obtained by decomposing the velocity into a mean and a fluctuating component, i.e. $V_i = v_i + v_i'$. The mean is taken over time, such that

$$v(x) = \frac{1}{T}\int_0^T V(x,t)\,\mathrm{d}t\,. \tag{2.6}$$

A similar approach is used for the pressure. The equations resulting from substituting the decomposed velocity and pressure into (2.1) are the Reynolds-averaged equations and can be written as

$$\frac{\partial v_i}{\partial t} + v_j\frac{\partial v_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) - \frac{\partial(\overline{v_i' v_j'})}{\partial x_j}. \tag{2.7}$$

The RANS equations are similar to the NS equations, with the exception of the extra term including the Reynolds stress tensor $R_{ij} = \overline{v_i' v_j'}$. This term originates from the mean of a fluctuating quantity being zero (i.e. $\overline{v'} = 0$), but this not being true in general for the mean of the product of two fluctuating terms (i.e. $\overline{v'v'}$). Therefore, the main goal of turbulence models is to model the Reynolds stress tensor to close the equations. The isotropic and deviatoric parts of the Reynolds stress tensor can be separated as

$$R_{ij} = \frac{2}{3}k\delta_{ij} + a_{ij}, \tag{2.8}$$

where $k \equiv \frac{1}{2}\overline{v_i' v_i'}$ is the turbulent kinetic energy. The anisotropic part of the Reynolds stress tensor can be non-dimensionalized as

$$b_{ij} \equiv \frac{a_{ij}}{2k}. \tag{2.9}$$

For the remainder of this work, this tensor will be referred to as the anisotropy tensor.

Eddy viscosity models Eddy viscosity models relate the turbulent stresses to the mean velocity gradients, introducing a proportionality constant ν_t called the eddy viscosity. When the Boussinesq assumption is used, the anisotropy tensor a_{ij} is related to the mean strain rate tensor S_{ij},

$$a_{ij} \approx -2\nu_t S_{ij}, \tag{2.10}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right), \tag{2.11}$$

in incompressible flows. There are various models for calculating the eddy viscosity, often classified by the number of additional partial differential equations that are introduced.

The most simple models are algebraic models. In these models, no additional PDEs are solved, and the eddy viscosity is calculated using a specified turbulent mixing length. This length is analogous the mean free path of molecules in a gas [163]. An advantage of these models is that they are simple to implement. However, they are generally only accurate for simple flows against which they have been calibrated, as the value of the mixing length depends on the flow under consideration. Two well-known algebraic turbulence models are the Baldwin-Lomax model [7] and the Cebeci-Smith model [19]. In one-equation models, only one additional PDE for a turbulent quantity is solved. An example is the Spalart-Allmaras turbulence model [140], which solves for a modified eddy viscosity. A consequence of solving only one additional PDE is that these models often have good numerical stability and have a low computational cost.

Two-equation models generally solve for the turbulent kinetic energy and a quantity related to a turbulent length- or time-scale. The formulation of the k-equation usually starts from the exact equation of the turbulent kinetic energy [90], as mainly the diffusion term needs modeling. The other equation (e.g. for ε or ω) can be derived exactly [147], but contain terms that are difficult to model. Therefore, these equations are usually derived to have a similar form as the k-equation, using dimensional and intuitive arguments. Examples of two-equation models are the $k-\varepsilon$ model [74] and the $k-\omega$ model [162, 163]. The main shortcoming of the $k-\varepsilon$ model is its lack of sensitivity to adverse pressure gradients [88]. In these circumstances, the shear-stress levels it predicts are too high. This delays or prevents the onset of separation. In low Reynolds number models, the equations are integrated through the viscous sublayer. An example is the Launder-Sharma model [74], which uses damping functions near the wall and solves for a modified version of ε. A common problem with these low Reynolds number models is their numerical stiffness. The $k-\omega$ model generally performs better under adverse pressure gradients than the $k-\varepsilon$ model. Furthermore, it can be applied in a low Reynolds number setting without modifications, as it does not use damping functions and has Dirichlet boundary conditions at the wall. This also results in better numerical stability. An important shortcoming of the $k-\omega$ model is its dependence on the freestream value of ω [89]. Many corrections to both models have been proposed by several authors.

Advantages of the Boussinesq approximation are that it has a simple form and that it has relatively good performance in flows where the mean velocity gradients change relatively slowly. Examples of these flows include the mixing layer, round jet, channel flow, and boundary layers. However, these advantages come at a significant cost. The Boussinesq approximation assumes a local relationship between the Reynolds stresses and the rate of strain. Even simple experiments can show that this is not generally the case. For example, the experiment performed by Uberoi [153] shows that the Boussinesq assumption predicts a constant or zero Reynolds stress tensor in sections where the experiment shows that it varies significantly. This also follows from the Reynolds stress equations, which show that the stresses are convected by both the mean and the fluctuating velocities [115]. Additionally, the Boussinesq approximation assumes that the principal axes of the Reynolds stress tensor and those of the strain rate tensor coincide. As a result, the eddy viscosity depends on a variety of factors, e.g. the characteristics of the boundary, flow history effects, and the freestream turbulence intensity. Examples of flows where the Boussinesq approximation fails are curved surfaces, rotating flows, and ducts with secondary motions [163].

Reynolds stress models A different approach is taken in Reynolds stress models. In this class of models, no eddy viscosity is used. Instead, transport equations for the Reynolds stress tensor are solved. As the Reynolds stress tensor is symmetric, this involves solving six coupled partial differential equations. One additional transport equation is solved in order to obtain a length- or time-scale [116, p. 387]. The transport equations for the Reynolds stresses are given by

$$\frac{\mathrm{D}}{\mathrm{D}t}\overline{v_i v_j} + \frac{\partial \mathcal{T}_{kij}}{\partial x_k} = \mathcal{P}_{ij} + \mathcal{R}_{ij} - \varepsilon_{ij}, \tag{2.12}$$

where $\mathcal{T}_{kij}$ is the Reynolds-stress flux, $\mathcal{P}_{ij}$ is the production tensor, $\mathcal{R}_{ij}$ is the pressure-rate-of-strain tensor, and ε_{ij} is the dissipation tensor. The convection and the production term are in closed form, but approximations are required for closing the dissipation tensor, pressure-rate-of-strain tensor, and the Reynolds stress flux.

The most crucial tensor from a modeling point of view is the pressure-rate-of-strain tensor, which is given as

$$\mathcal{R}_{ij} = \overline{\frac{p'}{\rho}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)}. \tag{2.13}$$

This tensor has zero trace, and redistributes the energy between the different components of the stress tensor. It can be decomposed into three parts with different characteristics: the rapid part, the slow part, and the harmonic part. The rapid part responds immediately to changes in the mean velocity gradients. The slow part, on the other hand, describes the relaxation to the isotropic equilibrium state. It can, for example, be modeled using the model proposed by Rotta [128]. Examples of models for the rapid term are the models by Launder et al. [75] or Speziale et al. [141]. At high Reynolds numbers, the dissipation tensor ε_{ij} can be modeled as an isotropic tensor due to local isotropy. This is less accurate at moderate Reynolds numbers. However, the anisotropic part of this tensor can be absorbed into the model for $\mathcal{R}_{ij}$. Finally, the Reynolds-stress flux $\mathcal{T}_{ijk}$ can be decomposed into three fluxes: viscous diffusion, the pressure transport, and turbulent convection. The viscous diffusion is in closed form, and is negligible outside of the viscous wall region. The pressure transport is low near the walls, and is usually neglected or absorbed into the model for the turbulent convection using a gradient-diffusion assumption. The turbulent convection is usually modeled assuming a gradient transport process, using k, ε, and the Reynolds stresses.

An advantage of Reynolds stress models is that they are more general than eddy viscosity models, as the turbulent viscosity assumption is not needed. The presence of convection and diffusion terms in the transport equations for the Reynolds stress tensor naturally takes into account some of the flow history, which is not accounted for in eddy viscosity models. Therefore, Reynolds stress models are expected to perform better in flows with high anisotropy and non-local effects. However, due to the additional transport equations these models have higher computational requirements and are harder to implement [91]. Furthermore, they generally have worse convergence properties than eddy viscosity models.

Nonlinear eddy viscosity models In an attempt to combine the improved generality of Reynolds stress models with the numerical robustness of eddy viscosity models, approximations can be made to the transport terms in the Reynolds stress tensor equations to reduce them to a set of algebraic equations. This results in a set of six algebraic equations in which the Reynolds stresses are related locally to the mean velocity gradients, k, and ε. A model resulting from this derivation is called an algebraic Reynolds stress model (ARSM). One example is the work of Rodi [127]. However, in this approach, the Reynolds stress is related implicitly to the mean velocity gradients, which can cause problems with the numerical stiffness. In this case, the advantage over solving the Reynolds stress transport equations is lost. However, additional assumptions can be made to make the relation between the Reynolds stress and the mean velocity gradients explicit. Such an explicit model is called an explicit algebraic Reynolds stress model (EARSM). These models can be seen as higher-order eddy viscosity models. One of the earliest attempts was by Pope [115], who formulated a more general relation between the mean velocity gradients and the stress tensor. Assuming that the transport terms in the Reynolds stress equations are negligible, it was shown that at sufficiently high Reynolds numbers the Reynolds stress tensor can be expressed as a tensor polynomial with a finite number of terms. Another example is the work of Craft et al. [24], who formulated a cubic eddy viscosity model based on the model by Wallin and Johansson [154]. One major challenge in developing these algebraic models is determining the closure coefficients. One approach is calibrating these coefficients using a set of basic flows [133]. An alternative approach is to derive the values from the implicit relation from which the explicit model is derived [154]. An advantage of this class of turbulence models is thus the numerical robustness resulting from the explicit relation between the Reynolds stress and the mean velocity gradients. However, these numerical properties come at the cost of a loss of generality compared to RSMs. For example, the non-local effects taken into account in RSMs are not represented in EARSMs.

2.3. Representation of the anisotropy tensor

As will be discussed later, several data-driven approaches to turbulence modeling apply corrections to the anisotropy tensor, or predict it directly. However, finding a suitable representation for (corrections to) the anisotropy tensor remains a challenge and is a topic of current research (e.g. Wu et al. [165]). In this section, the properties of the Reynolds stress tensor and the anisotropy tensor are described. Furthermore, the barycentric map is introduced, which is used for interpreting the eigenvalues and thus the realizable states of a given turbulence model. Finally, several methods for correcting the eigenvectors are discussed and compared.

2.3.1. Properties of the Reynolds stress and anisotropy tensor

A symmetric $n \times n$ real matrix $\boldsymbol{A}$ is positive semi-definite if

$$\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \geq 0, \tag{2.14}$$

for any non-zero column vector $\boldsymbol{x}$ of n real numbers. The Reynolds stress tensor is constructed by taking the outer product of the fluctuating velocity with itself,

$$\boldsymbol{v}' \otimes \boldsymbol{v}' = \begin{bmatrix} v_1' \\ v_2' \\ v_3' \end{bmatrix} \begin{bmatrix} v_1' & v_2' & v_3' \end{bmatrix} = \begin{bmatrix} v_1'v_1' & v_1'v_2' & v_1'v_3' \\ v_2'v_1' & v_2'v_2' & v_2'v_3' \\ v_3'v_1' & v_3'v_2' & v_3'v_3' \end{bmatrix} = v_i'v_j', \tag{2.15}$$

and applying Reynolds averaging. This matrix is always positive semi-definite, as

$$\boldsymbol{x}^T \boldsymbol{v}'\boldsymbol{v}'^T \boldsymbol{x} = (\boldsymbol{x}^T \boldsymbol{v}')^2 \geq 0. \tag{2.16}$$

This is still true after applying Reynolds averaging, as all samples satisfy this property. This positive semi-definiteness guarantees that the eigenvalues are non-negative. Following from Schwartz' inequality and the fact that the velocities are real, Schumann [131] described that

$$\overline{v_\mu' v_\mu'} \geq 0, \qquad \overline{v_\mu' v_\mu'} + \overline{v_\nu' v_\nu'} \geq 2|\overline{v_\mu' v_\nu'}|, \qquad \text{and} \qquad \det\left(\overline{v_i' v_j'}\right) \geq 0, \tag{2.17}$$

for $\mu, \nu \in \{1,2,3\}$. The minimum and maximum values of $\overline{v_\mu' v_\mu'}$ are 0 and $2k$, respectively, where the maximum bound follows from the definition of k.

From its definition, the Reynolds stress tensor is symmetric and therefore diagonalizable. Separating the Reynolds stress tensor into its isotropic and deviatoric part and using the definition of the anisotropy tensor yields

$$R_{ij} = 2k\left(\frac{\delta_{ij}}{3} + b_{ij}\right) = 2k\left(\frac{\delta_{ij}}{3} + V_{ik}\Lambda_{kl}V_{jl}\right), \tag{2.18}$$

where V_{ij} is a matrix of which the columns are the eigenvectors of the anisotropy tensor, and $\Lambda \equiv \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ is a diagonal matrix with the eigenvalues of the anisotropy tensor on the diagonal. This representation of the Reynolds stress separates its magnitude (the turbulent kinetic energy), shape (the eigenvalues), and orientation (the eigenvectors). The eigenvectors define a coordinate system called the principal coordinate system. In this coordinate system, the Reynolds stress tensor is diagonal. From the bounds on the diagonal elements of the Reynolds stress tensor discussed earlier, its eigenvalues ϕ_i are bounded by

$$0 \leq \phi_i \leq 2k. \tag{2.19}$$

These eigenvalues are related to the eigenvalues of the anisotropy tensor by

$$\lambda_i = \frac{\phi_i}{2k} - \frac{1}{3}. \tag{2.20}$$

From the definition of the anisotropy tensor, it can be derived that

$$-\frac{1}{3} \leq b_{\mu\mu} \leq \frac{2}{3}, \qquad -\frac{1}{2} \leq b_{\mu\nu} \leq \frac{1}{2}, \qquad \mu \neq \nu. \tag{2.21}$$

An thus the eigenvalues of the anisotropy tensor are bounded by $-1/3 \leq \lambda_i \leq 2/3$.

2.3.2. Eigenvalues

Significant work has been performed for the representation of the eigenvalues. The three invariants of the anisotropy tensor were discussed by Lumley [84], and are given by

$$\text{I} = b_{ii}, \qquad \text{II} = b_{ij}b_{ji}, \qquad \text{III} = b_{ij}b_{in}b_{jn}. \tag{2.22}$$

Banerjee et al. [8] represented the anisotropy tensor as a convex combination of one, two, and three component limiting states of turbulence:

$$b_{ij} = C_{1c}b_{1c} + C_{2c}b_{2c} + C_{3c}b_{3c}. \tag{2.23}$$

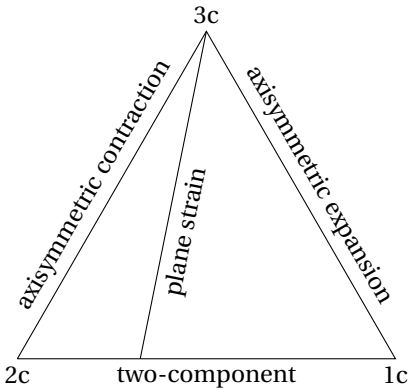


Figure 2.1: Barycentric map.

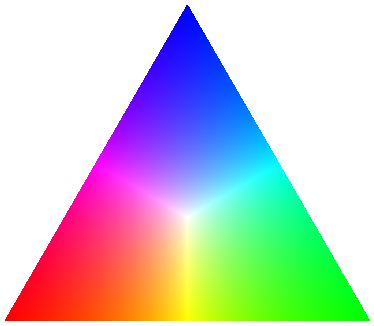


Figure 2.2: RGB-colormap legend.

with $C_{1c} + C_{2c} + C_{3c} = 1$ and $C_{1c} \geq 0$, $C_{2c} \geq 0$, and $C_{3c} \geq 0$. Componentiality in this case refers to the number of non-zero velocity fluctuations. The barycentric coefficients are then related to the eigenvalues by

$$\begin{aligned} C_{1c} &= \lambda_1 - \lambda_2 \\ C_{2c} &= 2(\lambda_2 - \lambda_3) \\ C_{3c} &= 3\lambda_3 + 1, \end{aligned} \tag{2.24}$$

where the eigenvalues are ordered according to $\lambda_1 \geq \lambda_2 \geq \lambda_3$. As the anisotropy tensor is represented as a convex combination of limiting states, it is possible to map it to a position in a map defined by three vertices (ξ_1, η_1), (ξ_2, η_2), and (ξ_3, η_3), displayed in fig. 2.1. The corners of the barycentric map correspond to the one-, two-, and three-component limiting states, as indicated. The eigenvalues can then be bounded by considering their corresponding position in the barycentric map. Furthermore, corrections or perturbations to the eigenvalues can be represented in terms of two barycentric coordinates.

2.3.3. Eigenvectors

Finding a suitable representation for the correction to the eigenvectors is a challenging task. First of all, this is because, it is more difficult to bound the corrections to the eigenvectors, contrary to the eigenvalues. Thompson et al. [148] used the Reynolds stress transport equations to constrain the eigenvectors based on the eigenvalue bounds. Iaccarino et al. [55] tried to bound the eigenvectors using the value of the production term. Wang et al. [156] used Euler angles to represent the perturbations in the eigenvectors. Secondly, in order for the corrected anisotropy tensor to be realizable, the corrected set of eigenvectors should still be orthonormal. One way to ensure this is by specifying the correction as a three-dimensional rigid body rotation. Finally, the representation of the eigenvector corrections should be frame-independent and smooth. Wu et al. [166] compares three methods for representing the corrections to the eigenvectors as a three-dimensional rigid body rotation based on their invariance and smoothness properties. The first two representations use Euler angles. Given a local coordinate system $x - y - z$ which is initially aligned with the global coordinate system $X - Y - Z$, the orientation can be represented as a rotation about the z-axis with angle ϕ_1, a rotation about the x-axis with angle ϕ_2, and another rotation around the z-axis with angle ϕ_3. The first representation is the discrepancy in the absolute Euler angles representing the orientation of the original set of eigenvectors and the corrected set of eigenvectors. This representation is non-smooth and depends on the reference frame. The second property can be alleviated by specifying the discrepancy itself as Euler angles. However, for this representation non-smoothness remains an issue [166]. The third representation defines the discrepancy between two sets of eigenvectors as a rotation axis defined by the unit vector $\mathbf{n} = [n_1, n_2, n_3]$ and a rotation angle θ. This rotation can be represented using a unit quaternion, defined as

$$\mathbf{q} = \left[\cos\frac{\theta}{2}, n_1 \sin\frac{\theta}{2}, n_2 \sin\frac{\theta}{2}, n_3 \sin\frac{\theta}{2}\right]^T. \tag{2.25}$$

As $\mathbf{n}$ is a unit vector, then it is clear that $||\mathbf{q}|| = 1$. The representation of eigenvector discrepancies using unit quaternions is frame independent and smoother than the relative Euler angles.

2.4. Data-driven approaches to RANS turbulence modeling

The increase in computational resources and developments in experimental techniques have caused a rise in the availability of high-fidelity data. In combination with the improvements in the effectiveness of machine learning techniques, this enabled a more data-driven approach to complement the previously largely physics-based approach to turbulence modeling [28]. Machine learning was first used in turbulence modeling in research aiming to identify and visualize flow structures in turbulent channel flows [85] and to reconstruct these flow features near the walls [92]. For RANS simulations specifically, the first steps towards data-driven turbulence modeling have been set by using optimization and Bayesian inference approaches to calibrate the RANS coefficients [20, 33, 78, 101]. An important shortcoming of these approaches is that they only address the uncertainty in the RANS coefficients, and thus do not take into account the uncertainties introduced due to the specific form of the closure model. Also, these methods are highly problem-specific, i.e. they cannot be used for other flows than the ones they were analyzed for. This drawback has been slightly alleviated by using Bayesian model-scenario averaging [32, 33], but this approach still does not allow for using several different cases sharing similar flow characteristics. An alternative approach was taken by Dow and Wang [26], who used the adjoint method to infer the eddy viscosity field from DNS data and modeled the discrepancy as a Gaussian random field.

The state of the art of machine learning in turbulence modeling can roughly be divided into three categories. Firstly, Weatheritt and Sandberg [159, 160] use symbolic regression and gene expression programming to obtain tangible analytical expressions for the Reynolds stress anisotropy tensor. An advantage of this approach is that these expressions provide valuable insight into the turbulence model and result in an interpretable model. However, this might not be the case for more complex flow geometries [165].

Secondly, Ling et al. [82] and Wang et al. [157] directly predict the Reynolds stress tensor or its discrepancy from the data. There are several ways in which the Reynolds stress tensor can be corrected. Tracey et al. [151] used kernel regression to model only the eigenvalues of the stress tensor. Predicting the complete Reynolds stress tensor corresponds to predicting the eigenvectors in addition to the eigenvalues. This is done by Ling et al. [82], using a neural network architecture ensuring Galilean invariance [81]. The description of the Reynolds stress tensor is complete if also the prediction of the turbulent kinetic energy is taken into account, as is done by Wang et al. [158]. However, when the turbulent kinetic energy is also predicted by the machine learning algorithm, its transport equation and the physical reasoning behind it are not used at all. This reduces the algorithm even more to a black-box model. One disadvantage of this group of approaches is that the anisotropy tensor is needed from the data. It therefore depends on having accurate datasets from DNS or highly-resolved LES simulations, which are not widely available for high-Reynolds number flows.

Thirdly, Singh et al. [137, 138] try to obtain a more generalizable method by training machine learning algorithms to predict corrections applied to existing closure models. They follow the approach proposed by Duraisamy et al. [29] and Tracey et al. [152], which combines machine learning with statistical inference. In this approach, a spatially varying corrective term is inferred from a number of problems. Machine learning is then used to find the relation between the machine learning features and the database of inferred functions. This paradigm is called field inversion and machine learning (FIML). Subsequent work has further developed the paradigm [106, 107, 135, 136, 138, 171], and compared the performance of neural networks and (multiscale) Gaussian processes to predict the inferred corrective term for several turbulence models and flow geometries. There are several advantages to this paradigm. First of all, a relatively low number of inferred fields are necessary to provide training data for the machine learning algorithm [28]. Secondly, the inferred corrective term in itself provides valuable information to turbulence modellers. Finally, this approach can also be applied when the observational data are sparse or noisy. This paradigm is the main focus of this work and is further introduced in the next section.

2.5. The paradigm of field inversion and machine learning

As introduced in the previous section, the paradigm of field inversion and machine learning combines statistical inversion and machine learning. The first step is to apply a corrective function to a given forward model. This forward model can in general be any physical system defined by a set of governing equations and boundary conditions. In this work, the forward model corresponds to the RANS equations closed by the $k-\omega$ turbulence model. The corrective function $\boldsymbol{\beta}$ is then applied to a point in the model aiming to directly address the model-form errors. It should be stressed that this corrective term can vary spatially and temporally, and thus moves beyond merely tuning the closure coefficients. The choice of where to apply the corrective term is flexible. For example, in this work it is applied to the production term in the ω-equation of

the $k-\omega$ model and to the eigenvalues of the anisotropy tensor. Another example is the work of Tracey et al. [152], where the corrective term is formulated as the source term in the Spalart-Allmaras turbulence model.

The spatially varying corrective function is found by a high-dimensional optimization problem aiming to match a quantity of interest with that of a dataset. The discrepancy between the outcome of the augmented forward model and the data is quantified by the objective function J. As for the definition of the corrective term, there are many options for the choice of the objective function and the data. For example, in this work the sum of squared differences in the mean velocity of the augmented $k-\omega$ model and DNS or LES data is taken as the objective function. Duraisamy et al. [29] defined the difference in the wall shear stress as the objective function. The objective function is thus chosen based on the available quantities in the dataset or on the importance of a given quantity in the problem at hand. The data can come from high-fidelity simulations such as highly resolved LES or DNS or from experimental simulations. Furthermore, the data can be spatially distributed (e.g. a mean velocity field from a DNS) or a single or low number of scalar values (e.g. a force measurement from an experimental study).

Now that the forward model, the correction applied to it, and the objective functions are defined, it is necessary to find the corrective term which optimizes the objective function. This can be formulated as a field inversion problem, where the value of the corrective term in each grid cell can be found to optimize the objective function. This can be done for a variety of problems, resulting in a dataset of optimized corrective terms. This concludes the first phase of the paradigm, which is visualized in the flowchart in fig. 2.3 by phase 1. In each iteration of the optimization, the outputs of the augmented RANS model are compared to the data yielding a value of the objective function. The optimization algorithm then selects a new corrective term, which is used in the augmented RANS model to yield updated outputs. This process is iterated until convergence for several flow cases.

The second phase of the paradigm, the machine learning phase, aims to reconstruct the corrective term on unseen flow cases. This phase can be divided into a training phase and a prediction phase, which are also indicated in fig. 2.3. In the training phase, indicated by phase 2, a machine learning algorithm is used to approximate a function between the input features and the corrective term. In the context of turbulence modeling, input features are mean flow quantities which are available during the RANS simulation. For example, this feature set can consist of combinations of the strain and rotation rate tensor or physical quantities such as the wall-distance based Reynolds number of the streamline curvature. The dataset thus consists of the features and corrective terms resulting from the field inversion phase for a number of flow cases. In the prediction phase, the trained machine learning algorithm is used to predict the corrective term on unseen flow cases, given the input features. This can be implemented in an iterative setting, where in each iteration the flow solver provides the mean flow features to the machine learning algorithm. The algorithm then predicts the corrective term which is used in the flow solver to obtain an update of the features. This phase can also be implemented as a single correction, in what is called the corrective approach.

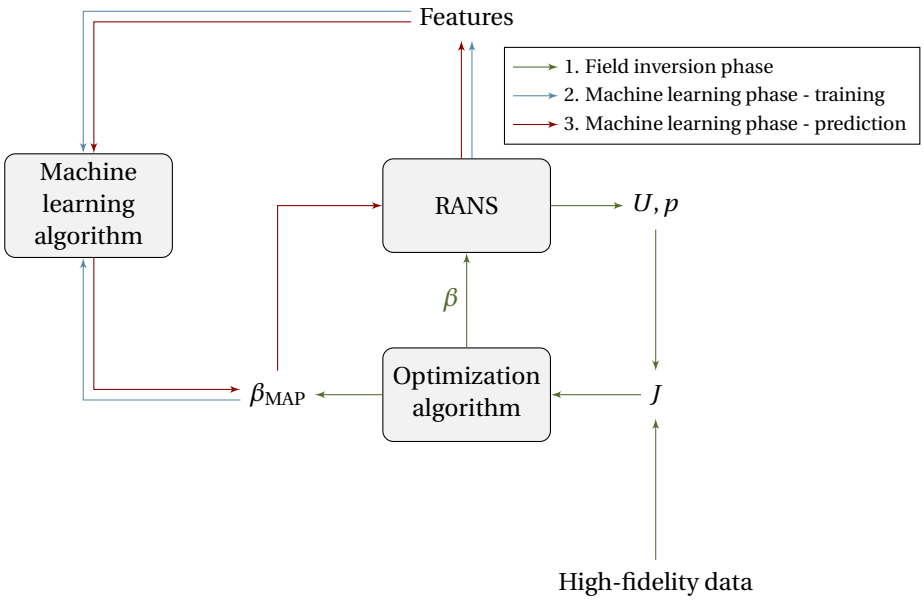


Figure 2.3: Flowchart of the paradigm of field inversion and machine learning.

3

Field inversion

This chapter provides the theoretical basis for the first part of the paradigm: the field inversion. Section 3.1 poses the problem statement of the field inversion phase. Section 3.2 describes the various optimization techniques, gives an overview of their applications in CFD, and argues why gradient-based optimization is the most relevant option for the problem at hand. An overview of gradient-based optimization algorithms and methods to obtain the gradients are discussed in section 3.3. Furthermore, it justifies the need for adjoint methodology for the computation of the gradient, which is further elaborated upon in section 3.4. Finally, the theory given in this chapter will be illustrated using a simple model problem in section 3.5.

3.1. Problem formulation

The discussion in this chapter is based on the theory presented by Aster et al. [4], unless stated otherwise. Broadly speaking, the goal of the field inversion phase is to find the corrective field of which, when it is used in the RANS solver, the mean flow best approximates the high-fidelity data. More formally, the forward model is denoted by $h(\boldsymbol{\beta})$. Here, $\boldsymbol{\beta}$ is a spatially varying correction to the forward model. It is assumed that the data $\boldsymbol{d}$ consists of the sum of the true data $\boldsymbol{d}_{\text{true}}$ and measurement noise ε. Then, the goal of the field inversion is to obtain $\boldsymbol{\beta}_{\text{true}}$ from the data, such that $h(\boldsymbol{\beta}_{\text{true}}) = \boldsymbol{d}_{\text{true}}$. In this work, the forward model corresponds to the RANS model augmented with the corrective function and the data corresponds to the high-fidelity (LES, DNS, or experimental) data.

One approach is to assume that the data errors are independent and normally distributed and to use the maximum likelihood principle to find the corrective field which maximizes the probability of the data given the corrective field. It can be easily shown that this approach corresponds to a least-squares problem, where the objective is to minimize the 2-norm of the difference between the forward model output and the data, i.e.

$$\boldsymbol{\beta}_{\text{MLE}} = \arg\min_{\boldsymbol{\beta}} ||h(\boldsymbol{\beta}) - \boldsymbol{d}||^2. \tag{3.1}$$

This approach comes with several challenges. First of all, if the forward model is non-linear, there might be several local optima. Secondly, $\boldsymbol{\beta}_{\text{true}}$ might not exist, for example because of an incorrectly specified model or because of noise in the data. Thirdly, there might be multiple corrective fields which give an output corresponding to the data. In this case, the solution to the inverse problem is not unique. Finally, the problem might be ill-conditioned, i.e. it is possible that a small change in the data causes a large change in the resulting model. These last two issues can be alleviated by adding a regularization term to the objective function, for example Tikhonov regularization:

$$\boldsymbol{\beta}_{\text{MLE}} = \arg\min_{\boldsymbol{\beta}} ||h(\boldsymbol{\beta}) - \boldsymbol{d}||^2 + \alpha||\boldsymbol{\beta} - \boldsymbol{\beta}_0||^2, \tag{3.2}$$

where α is the regularization parameter. The regularization term punishes values of $\boldsymbol{\beta}$ deviating too much from $\boldsymbol{\beta}_0$. The regularization parameter weighs the relative importance of fitting the data and regularizing the corrective term. Using regularization can thus help improve the conditioning of an inverse problem. There are many types of regularization [11] and strategies for choosing the regularization parameter [48]. However, different choices of regularization generally lead to different results.

In this work, a Bayesian approach is adopted. This approach results in a probability distribution over corrective fields rather than a single corrective field, as is the case with the maximum likelihood approach. The posterior distribution over the corrective fields given the data is found using Bayes' rule:

$$p(\boldsymbol{\beta}|\boldsymbol{d}) = \frac{p(\boldsymbol{d}|\boldsymbol{\beta})\,p(\boldsymbol{\beta})}{p(\boldsymbol{d})}, \tag{3.3}$$

where $p(\boldsymbol{\beta})$ is the prior distribution and $p(\boldsymbol{d})$ is the evidence. Assuming that the data consists of the sum of the forward model output and Gaussian noise,

$$\boldsymbol{d} = h(\boldsymbol{\beta}) + \varepsilon, \tag{3.4}$$

where $\varepsilon \sim \mathcal{N}(0, C_m)$ and C_m is the observational covariance matrix. Then, the likelihood is normally distributed:

$$\boldsymbol{d}|\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{d} - h(\boldsymbol{\beta}), C_m). \tag{3.5}$$

Also, it is assumed that $\boldsymbol{\beta}$ is normally distributed,

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}_{\text{prior}}, C_\beta), \tag{3.6}$$

where $\boldsymbol{\beta}_{\text{prior}}$ is the prior mean and C_β is the prior covariance matrix. Then, the posterior is proportional to

$$p(\boldsymbol{\beta}|\boldsymbol{d}) \propto \exp\underbrace{\left(-\frac{1}{2}(\boldsymbol{d} - h(\boldsymbol{\beta}))^T C_m^{-1}(\boldsymbol{d} - h(\boldsymbol{\beta})) - \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}})^T C_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}})\right)}_{J}. \tag{3.7}$$

Maximizing the objective function J (or minimizing its negative) thus finds the maximum a posteriori (MAP) solution.

There are multiple options for choosing the observational covariance matrix. In the most simple option, the datapoints are assumed to be independent with constant variance, i.e. the posterior covariance matrix is a diagonal matrix with identical diagonal elements σ_m^2. An extension to this model would be to represent the observational covariance matrix by a diagonal matrix where each diagonal element contains the variance of each observation. The variance of each element can, for example, be calculated from the statistics of the dataset, given that multiple realizations of the same process are available. The only model in which covariance between multiple elements is taken into account is when a full observational covariance matrix is used. One example where this can be used is in particle image velocimetry (PIV), where the off-diagonal elements of the observational covariance matrix depend on the overlap and the interrogation window sizes [5, 161]. For example, if multiple realizations of the same dataset D are available, the exact covariance matrix is given by

$$C_m = \mathbb{E}\left[(D_i - \overline{D_i})(D_j - \overline{D_j})\right], \tag{3.8}$$

where $\overline{D_i} = \mathbb{E}[D_i]$.

There are many ways to select the prior distribution in Bayesian statistics [18]. In the context of field inversion, Parish and Duraisamy [107] suggest choosing a Gaussian prior with a constant diagonal prior covariance matrix and determining the variance using the following procedure:

1. Assume the variance σ_β^2.
2. Sample from the Gaussian prior with the assumed variance.
3. Propagate the sampled corrective terms through the forward model.
4. Check whether the data lies between the $\pm 2\sigma$ limits resulting from the propagated corrective term samples.
5. Adjust variance if necessary.

If the prior and the likelihood are specified using the simplest covariance matrices, i.e. $C_m = 1/\sigma_m^2 I$ and $C_\beta = 1/\sigma_\beta^2 I$, the objective function simplifies to

$$J = \frac{1}{\sigma_m^2}\sum_{i=1}^{N}(d_i - h(\beta)_i)^2 + \frac{1}{\sigma_\beta^2}(\beta_i - \beta_{\text{prior},i})^2, \tag{3.9}$$

where N is the number of grid cells. Optimizing this objective function corresponds to minimizing the sum of squares between the model output and the data, while punishing values of the corrective function which deviate too much from the prior. This corresponds to Tikhonov regularization with the equivalent regularization parameter equal to $\sigma_m^2/\sigma_\beta^2$. This parameter specifies the relative importance of the terms in the objective function. For lower values, it is expected that the data is a better observation of the truth, and thus the likelihood has a higher importance in the optimization problem than the prior, and vice versa.

There are several advantages to using a Bayesian approach. First of all, it allows for implementing prior information into the model through the prior distribution. For example, the prior mean can be chosen to be equal to the corrective term corresponding to the baseline RANS model. Secondly, the posterior distribution can be used to obtain an estimate of the uncertainty of the MAP solution. It is possible to approximate the posterior covariance matrix as the inverse of the Hessian of the objective function:

$$C_{\beta_{\text{MAP}}} = H^{-1}\big|_{\beta_{\text{MAP}}} = \left[\frac{\delta^2 J}{\delta\beta_i \delta\beta_j}\right]^{-1}\Bigg|_{\beta_{\text{MAP}}}. \tag{3.10}$$

Using the Cholesky decomposition of the posterior covariance matrix,

$$RR^T = C_{\beta_{\text{MAP}}}, \tag{3.11}$$

it is then possible to sample from the posterior distribution,

$$\beta = \beta_{\text{MAP}} + Rs, \tag{3.12}$$

where s is a vector of independent standard normal variates. This is valid since

$$\text{var}[\beta] = \text{var}[\beta_{\text{MAP}} + Rs] = R\text{var}[s]R^T = RIR^T = C_{\beta_{\text{MAP}}}. \tag{3.13}$$

The sampled corrective terms from the posterior distribution can then be propagated through the forward model to obtain an estimate of the variance in the quantities of interest.

The number of samples needed for determining the prior and posterior variance can be found using an approximation of the standard error of the mean and variances. For n statistically independent samples, the standard error of the mean and variances are given by

$$\sigma_{\overline{x}} = \frac{\sigma}{\sqrt{n}} \qquad \text{and} \qquad \sigma_{S^2} = \sigma^2\sqrt{\frac{2}{n-1}}. \tag{3.14}$$

Depending on the computational cost of solving the forward model, these procedures can be rather costly.

As will be explained in section 3.4.3, the most efficient way to compute the Hessian of the objective function still requires N RANS simulations. Therefore, the Hessian is approximated. This work employs the approach taken by Singh et al. [136] and Singh and Duraisamy [135], where the Hessian is approximated using the Gauss–Newton approximation. If the objective function can be written as a sum of squares, i.e.

$$J(\beta) = \sum_{i=1}^{N} r_i^2(\beta), \tag{3.15}$$

then the second derivative can be derived and simplified according to the Gauss-Newton approximation:

$$\frac{\delta^2 J_i}{\delta\beta_j\delta\beta_k} = 2\sum_{i=1}^{m}\left[\frac{\delta r_i}{\delta\beta_j}\frac{\delta r_i}{\delta\beta_k} + r_i\frac{\delta^2 r_i}{\delta\beta_j\delta\beta_k}\right] \approx 2\sum_{i=1}^{m}\frac{\delta r_i}{\delta\beta_j}\frac{\delta r_i}{\delta\beta_k}. \tag{3.16}$$

The impact of this assumption is thus low if the first term dominates the second term. One case where this is true is when r_i is small, the *small-residual case* [98, p. 260]. In the context of the field inversion phase, r_i represents the misfit between the model output and the data. The Hessian is usually only required for the MAP solution, and thus near a (local) minimum where this misfit is expected to be small. Another case where the assumption holds is when the functions r_i are not highly nonlinear with respect to β, such that the second derivatives are small. The validity of this case is harder to quantify for nonlinear inverse problems, for example when the forward model is a CFD solver. The gradient in the remaining term in the Hessian approximation is constructed by running the adjoint solver considering only the i-th datapoint in the objective

function. This can be done relatively cheaply using a converged solution of the adjoint equations as initial condition.

In the derivation, it is assumed that the distributions are Gaussian. The impact of these assumptions can be quantified using Markov chain Monte Carlo sampling. This is done by Parish and Duraisamy [107] for the one-dimensional example problem. For that particular problem, the posterior agrees well with the MAP. However, this is not necessarily the case for other non-linear problems.

3.2. Optimization techniques

Having outlined the problem of field inversion in the context of RANS turbulence modeling, the algorithms used for the optimization of the objective function will now be discussed. This section gives an overview of the various classes of optimization algorithms available and argues why gradient-based algorithms are the most convenient choice for the problem at hand.

In gradient-based optimization methods, (an approximation of) the gradient is used to determine the direction to step into [98]. For the computation of the gradient, it is necessary that the objective function is sufficiently smooth. Furthermore, gradient-based methods only guarantee convergence when there exists a single global minimum or if the initial guess is sufficiently close to the global minimum, given an appropriate step size. Genetic algorithms, on the other hand, use an analogy of the theory of evolution to search for an optimal solution [9]. The optimization problem is parameterized into a set of genes, which are evaluated using a fitness function. For each population (i.e. a step in the optimization), the genes are manipulated by a number of processes. Genetic algorithms generally work for non-smooth objective functions with multiple local optima, and do not require gradients. Also, genetic algorithms lend themselves naturally to optimization problems which include discrete parameters, for example the number of engines on an aircraft. However, genetic algorithms suffer from relatively slow convergence. Generally, genetic algorithms use a significantly higher number of function evaluations than gradient-based optimization methods [99]. Also, the high-dimensional nature of the optimization problems considered in this paradigm would require a large population size, and therefore a large number of cost function evaluations [73]. Another approach to solve the inverse problem is to use an ensemble Kalman filter (EnKF) [56]. This statistical method has been applied in the scope of turbulence modeling to tune model parameters [41, 69]. A drawback of this approach is that it requires a large number of simulations, especially for high-dimensional problems [70].

Numerical optimization in computational fluid dynamics has mostly been discussed in the context of aerodynamic shape optimization, which has been studied extensively, using a variety of optimization methods. Examples of works where genetic algorithms are used include Poloni and Mosetti [114], Quagliarella and Della Cioppa [120], and Gage and Kroo [40]. Approximate objective function surface schemes have been used by Toropov [150], Giunta et al. [44], and Chiandussi [21]. Finally, gradient-based methods have been applied in a large number of works, for example in Narducci et al. [96], Baysal and Eleshaky [10], and Borggaard et al. [13].

The goal of the field inversion phase is to infer a corrective term which varies throughout the whole domain. The infinite-dimensional nature of this optimization problem leads to a high number of variables to be optimized in the discrete setting, with the dimensionality increasing with grid refinements. It could therefore be expected that, in the context of field inversion, gradient-based optimization algorithms will perform better than genetic algorithms and ensemble Kalman filters, given that there is an efficient way to obtain the gradient with respect to a high number of parameters. As will be discussed in the next section, adjoint methods provide a way to obtain the gradients of the cost function which is practically independent of the number of variables to be optimized. Therefore, the optimization problems considered in this work lend themselves most conveniently to gradient-based optimization, which will be the focus of the coming sections.

3.3. Gradient-based optimization

3.3.1. Optimization algorithms

The most simple gradient-based optimization algorithm is gradient descent. Gradient descent uses the gradient as the direction in which to change the parameters at step k with a given or calculated stepsize α^k. The i-th element of the corrective term is then updated according to

$$\beta_i^{k+1} = \beta_i^k + \alpha^k \frac{\partial J^k}{\partial \beta_i}. \tag{3.17}$$

This step size can either be chosen to be constant or calculated using a line search. Choosing the step size too small can slow down convergence, whereas choosing the step size too large can make the method fail to converge at all. If the step size is found using an exact line search, each step is orthogonal to the previous one. This results in a "zigzagging" behavior which can be inefficient, especially near a (local) optimum. This behavior can be reduced by including a momentum term [119]:

$$\beta_i^{k+1} = \beta_i^k + \alpha^k \frac{\partial J^k}{\partial \beta_i} + \mu^k (\beta_i^k - \beta_i^{k-1}), \tag{3.18}$$

where $0 \leq \mu^k \leq 1$. Gradient descent has linear convergence.

The conjugate gradient method improves upon gradient descent by taking into account the history of search directions. It requires subsequent search directions to be conjugate. For a quadratic objective function,

$$J = \frac{1}{2} \beta^T A \beta - b^T \beta, \tag{3.19}$$

a set of non-zero vectors $\{p_1, \ldots, p_n\}$ is conjugate with respect to A if

$$p_i^T A p_j = 0, \qquad \forall i \neq j. \tag{3.20}$$

It can be shown that this problem is equivalent to solving the system of linear equations $A\beta = b$, and that conjugate gradient solves it in n iterations. If A is a diagonal matrix, the contours of the objective function are ellipsoids with their axes aligned with the coordinate frame. Therefore, the optimization problem can be solved in n iterations by solving a univariate optimization problem along each axis. If A is not diagonal, these axes are not aligned. Introducing the conjugate directions corresponds to transforming the optimization problem to a coordinate system in which the contours are elliptical again, as the transformed matrix becomes diagonal due to conjugacy. The conjugate gradient method is generalized for non-linear optimization problems in the Fletcher–Reeves nonlinear conjugate gradient method [35].

In contrast with the previous methods, Newton's method uses the second-order derivative of the objective function. The corrective term is updated according to

$$\beta_i^{k+1} = \beta_i^k - \alpha^k \left[\frac{\partial^2 J^k}{\partial \beta_i \partial \beta_j} \right]^{-1} \frac{\partial J^k}{\partial \beta_i}. \tag{3.21}$$

The term within brackets is the Hessian. Newton's method converges quadratically. However, due to the calculation of the Hessian it is significantly more costly than first-order methods for high-dimensional problems. Furthermore, Newton's method can have problematic convergence far away from the optimum and when the Hessian is singular [98, p. 135]. Quasi-Newton methods trade off the cost of computing the Hessian and the increased convergence of quadratic methods by approximating the Hessian. Various methods can be identified which differ in the way the approximated Hessian is updated, e.g. the Davidon-Fletcher-Powell (DFP) method [25] or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [36]. Another approach aiming to address the weaknesses of Newton's method is used in trust region methods. These methods define a region in which the quadratic approximation is expected to be valid. This can result in an improved robustness compared to Newton's method [98, p. 262].

3.3.2. One-shot optimization

For optimization problems where the forward model is solved iteratively, there are several ways to implement the optimization algorithm. In the first approach, which will be referred to as the complete approach, the forward model is iterated until convergence, after which the corrective term is updated and the process is repeated. In the second approach, the corrective term is updated in each iteration of the forward model. This approach is referred to as the one-shot approach or tightly coupled optimization [144, 145].

The main reason for choosing the one-shot over the complete approach is that it can greatly reduce the computational cost of the full optimization problem, as the forward model is not iterated until convergence at each step of the optimization [144]. Using partially converged gradients can cause the optimization to diverge [43]. However, by starting the optimization from a converged solution, smoothing the gradients at each iteration, and choosing an appropriate step size, this problem can be alleviated to a large extent [61]. A possible practical advantage of the complete approach is that the forward model can be treated as a black box, as the iterative process does not have to be modified.

3.3.3. Sensitivity analysis

As noted before, gradient-based optimization methods require a calculation of the gradient of the objective function at each step in the optimization. Furthermore, it was mentioned that for high-dimensional optimization problems, the computation of the gradient can take up a large portion of the computational cost of the optimization. There are several ways to calculate the gradient of the objective function. A simple but naive approach would be to use finite differences, i.e.

$$\left.\frac{\delta J}{\delta \beta}\right|_{\beta_0} \approx \frac{J(\beta_0+\varepsilon)-J(\beta_0)}{\varepsilon} - \frac{\varepsilon}{2!}\left.\frac{\delta^2 J}{\delta \beta^2}\right|_{\beta_0}. \tag{3.22}$$

One disadvantage of this approach is that it requires a number of simulations equal to $N+1$. Even for the simplest two-dimensional flow geometries considered in this work, this approach is excessively expensive. Furthermore, the computed gradient is only an approximation, becoming more exact with decreasing perturbation magnitude. However, small perturbation magnitudes can lead to significant subtractive cancellation errors. This trade-off between sources of error requires the evaluation of the sensitivity for various perturbation magnitudes, and thus further increases the computational cost. One advantage of this approach is that it treats the forward model as a black-box. This means that only the output for a given input is needed for calculating the gradient.

Another approach to obtain the sensitivities is the complex variable method [142]. In this approach the objective function is expanded as a Taylor series with a complex increment, i.e.

$$J(\beta_0+i\varepsilon) \approx J(\beta_0) + i\varepsilon \left.\frac{\delta J}{\delta \beta}\right|_{\beta_0} - \frac{\varepsilon^2}{2!}\left.\frac{\delta^2 J}{\delta \beta^2}\right|_{\beta_0} - \frac{i\varepsilon^3}{3!}\left.\frac{\delta^3 J}{\delta \beta^3}\right|_{\beta_0}. \tag{3.23}$$

The gradient can then be approximated by considering the imaginary part of this expression:

$$\left.\frac{\delta J}{\delta \beta}\right|_{\beta_0} \approx \frac{\mathrm{Im}[J(\beta_0+i\varepsilon)]}{\varepsilon} + \frac{\varepsilon^2}{3!}\left.\frac{\delta^3 J}{\delta \beta^3}\right|_{\beta_0}. \tag{3.24}$$

This approach has the advantage that it does not suffer from subtractive cancellation errors. However, the program does need to be adjusted for accepting complex arguments in this case, and can therefore not be treated as a black-box anymore. Furthermore, this approach still requires varying each of the optimization parameters separately.

The third approach is the adjoint method. This method allows the calculation of the sensitivity of the objective function at a computational cost which does not depend on the number of optimization variables. In the context of field inversion, this approach is by far the most efficient approach and will be explained in detail in the next section.

3.4. Adjoint methodology

In the adjoint approach, the calculation of the gradient of the cost function is decoupled from the sensitivities of the primal variables. As a result of this decoupling, the gradients can be obtained at a cost which is practically independent of the number of optimization parameters. In the context of computational fluid dynamics, the adjoint method has mainly been used in optimal design, for example in the works of Jameson [59], Baysal and Eleshaky [10], Anderson and Venkatakrishnan [3], and Pini et al. [112]. In this setting, one aims to optimize the lift-to-drag ratio, the pressure distribution, or another quantity of interest by optimizing a (large number of) design variables parameterizing the geometry. Other applications include error analysis [64, 124], mesh adaptation [164], uncertainty quantification [31], and data assimilation [50]. There are two main variants of the adjoint methodology: the discrete and the continuous adjoint [43]. In the discrete approach, the governing equations are first discretized, after which the adjoint equations are derived. In the continuous approach, the adjoint equations are first derived in continuous form and then discretized.

3.4.1. Discrete adjoint

In this discussion, $i, j \in [1, N]$ and $k, m, n \in [1, M]$, where N is the number of design variables and M is the product of the number of grid nodes and the number of equations per node, following the derivation and notation of Papadimitriou and Giannakoglou [104]. Let $J(U(\beta), \beta)$ be an integral quantity of interest, in this work the objective function of the optimization problem. The functional depends both on the primal variables U and a set of parameters β. In this work, the primal variables are the mean velocity, pressure, and turbulence

model variables and β is the corrective term. The gradient of the objective function can be written using the chain rule:

$$\frac{\mathrm{d}J}{\mathrm{d}\beta_i} = \frac{\partial J}{\partial \beta_i} + \frac{\partial J}{\partial U_k}\frac{\mathrm{d}U_k}{\mathrm{d}\beta_i}. \tag{3.25}$$

The partial derivatives can be derived directly from the definition of the objective function. Also, the set of governing equations will be denoted as

$$R_m(U,\beta) = 0. \tag{3.26}$$

As changing the corrective term should not change the validity of the governing equations, the derivatives of the governing equations with respect to the corrective term are also zero:

$$\frac{\mathrm{d}R_m}{\mathrm{d}\beta_i} = \frac{\partial R_m}{\partial \beta_i} + \frac{\partial R_m}{\partial U_k}\frac{\mathrm{d}U_k}{\mathrm{d}\beta_i} = 0. \tag{3.27}$$

Again, the partial derivatives follow straightforwardly from the discretization of the governing equations. The sensitivity of the objective function can be obtained by solving (3.27) for $\mathrm{d}U_k/\mathrm{d}\beta_i$ and substituting it in equation (3.25). This corresponds to the direct approach, and requires N system solves. The adjoint approach multiplies the gradient of the residuals with a Lagrange multiplier and adds it to the expression of the gradient of the functional, thereby obtaining the augmented functional L:

$$\frac{\mathrm{d}L}{\mathrm{d}\beta_i} = \frac{\partial J}{\partial \beta_i} + \frac{\partial J}{\partial U_k}\frac{\mathrm{d}U_k}{\mathrm{d}\beta_i} + \Psi^T\left(\frac{\partial R_m}{\partial \beta_i} + \frac{\partial R_m}{\partial U_k}\frac{\mathrm{d}U_k}{\mathrm{d}\beta_i}\right) \tag{3.28}$$

$$= \frac{\partial J}{\partial \beta_i} + \Psi^T\frac{\partial R_m}{\partial \beta_i} + \left(\frac{\partial J}{\partial U_k} + \Psi^T\frac{\partial R_m}{\partial U_k}\right)\frac{\mathrm{d}U_k}{\mathrm{d}\beta_i}. \tag{3.29}$$

The adjoint equations are obtained by zeroing the expression between brackets,

$$R_k^\Psi = \frac{\partial J}{\partial U_k} + \Psi^T\frac{\partial R_m}{\partial U_k} = 0, \tag{3.30}$$

thereby removing the need to calculate the sensitivity of the primal variables with respect to the corrective term. Solving the adjoint equations requires one system solve. The obtained adjoint variable can then be used to calculate the sensitivity according to

$$\frac{\mathrm{d}L}{\mathrm{d}\beta_i} = \frac{\partial J}{\partial \beta_i} + \Psi^T\frac{\partial R_m}{\partial \beta_i}. \tag{3.31}$$

The calculation of the sensitivity of the functional is now reduced to one system solve, and is thus independent of the number of optimization variables.

3.4.2. Continuous adjoint

In the continuous approach, the adjoint equations are first derived in continuous form and then discretized. The first appearance of the continuous adjoint in the context of optimal design was in the work of Pironneau [113]. Further introductions to the theory behind the continuous adjoint are given by Giles and Pierce [43] and Jameson [60]. The derivation follows a similar procedure as that of the discrete adjoint. The adjoint variables are introduced as Lagrange multipliers and the terms which depend on the derivatives of the primal variables with respect to the corrective term are grouped. More precisely, the derivative of the augmented objective function can be written as

$$\frac{\delta L}{\delta \beta} = \frac{\delta J}{\delta \beta} + \int_\Omega \Psi\frac{\delta R}{\delta \beta}\,\mathrm{d}\Omega. \tag{3.32}$$

Again, the derivative of the objective function can be rewritten using the chain rule and the resulting partial derivatives can be derived straightforwardly from the definition of the objective function. By (repeatedly) applying integration by parts and collecting the terms including the sensitivities of the primal variables with respect to the corrective term, an expression of domain and boundary integrals is obtained. Equation (3.32) can then be rewritten as

$$\frac{\delta L}{\delta \beta} = \int_\Omega R^\Psi\frac{\delta U}{\delta \beta}\,\mathrm{d}\Omega + \int_\Gamma D^\Psi\left(\frac{\delta U}{\delta \beta}\right)\mathrm{d}\Gamma + G, \tag{3.33}$$

Approach	Required number of solver calls
Direct-direct	$N(N+1)/2+N$
Direct-adjoint	$N+1$
Adjoint-direct	$2N+1$
Adjoint-adjoint	$2N+1$

Table 3.1: Comparison of direct and adjoint approaches for computing the Hessian [104].

where G is the resulting expression for the gradient, consisting of all terms which do not include the sensitivity of the primal variables. By ensuring that

$$R^{\Psi} = 0 \qquad \text{and} \qquad D^{\Psi}\left(\frac{\delta U}{\delta \beta}\right) = 0, \tag{3.34}$$

the derivative of the augmented objective function does not explicitly depend on the derivatives of the primal variables. The adjoint equations and boundary conditions are then obtained from (3.34).

There are several restrictions on the choice of boundary conditions and objective function, which are derived by Giles et al. [42] and Jameson et al. [62]. The main restriction is that the part of the objective function which is specified on the boundary must be chosen such that the sensitivities of the primal variables in the boundary integral of (3.33) can be cancelled. In this work, the objective function is only specified in the domain, on which there are no restrictions [60]. Therefore, the restrictions pose no problems to the methodology considered here.

Continuous adjoint formulations have been derived for several common RANS turbulence models. The continuous adjoint for the incompressible and compressible Spalart-Allmaras turbulence model have been derived by Zymaris et al. [172] and Bueno-Orovio et al. [16], respectively. The continuous adjoint for the low-Re Launder-Sharma $k-\varepsilon$ turbulence model was derived by Papoutsis-Kiachagias et al. [105]. For the high-Re version, the continuous adjoint formulation was developed by Zymaris et al. [173], deriving adjoint wall functions for solving the adjoint equations. A hybrid approach is used by Taylor et al. [146], where the continuous adjoint is used for the flow equations, and the discrete adjoint is used for the turbulence model. Finally, for the $k-\omega$ SST model the continuous adjoint was derived by Kavvadias et al. [71]. Most of these studies include an investigation into the effect of assuming that the turbulence model variables do not depend on the design variables. This so-called *frozen turbulence assumption* is not relevant for this work, as the corrective term is applied to the turbulence model.

3.4.3. Hessian computation

So far, adjoint methods have only been considered as a means to obtain the first derivatives of the objective function. However, these methods can also be used to obtain the Hessian of the objective function with respect to the corrective term. In the context of field inversion, the Hessian is necessary if second order optimization algorithms are used in the field inversion phase. Also, its inverse can be used to approximate the posterior covariance matrix in order to obtain uncertainty bounds on the MAP solution [4].

There are four ways in which the Hessian can be obtained, corresponding to all combinations of using direct or adjoint methods to calculate the first and second derivatives. For the discrete adjoint, each approach is derived in [104] and the most efficient one is used in the Newton method in order to compare the convergence of the optimization with optimization methods using only first-order information. A similar study was performed in [103] for the continuous adjoint. The computational cost of the various approaches are compared in table 3.1, of which the direct-adjoint approach is the most efficient. However, it still requires a number of system solves proportional to the number of grid cells. Therefore, it is not expected to scale to the dimensions of the problems considered in this work, and approximations to the Hessian are expected to be more feasible computationally. Also, the problems concerning the derivation, implementation, and efficiency encountered in the adjoint for the first derivative become worse for higher derivatives [110].

3.4.4. Discussion

The theory introduced in the previous subsections raises the question of what approach to use for the field inversion. The relative advantages of using the discrete or continuous adjoint to obtain the sensitivity of the objective function have been discussed extensively, for example by Nadarajah and Jameson [95], Giles and Pierce [43], and Peter and Dwight [110]. One advantage of the discrete adjoint is that it provides the exact gradient of the discretized objective function, whereas the continuous approach approximates the continuous

gradient. Therefore, during an optimization using the discrete adjoint, the gradient information is consistent with the objective function evaluations. Giles and Pierce [43] argue that this difference is most relevant near a local optimum. Also, it makes verifying the discrete adjoint using finite differences more straightforward. Advantages of the continuous approach are that the physical meaning of the terms in the continuous adjoint equations is more clear and that the adjoint code is simpler and requires less memory. Also, the continuous approach generally lends itself more conveniently to the implementation in open-source CFD solvers (e.g. OpenFOAM [63]). On the other hand, the discrete adjoint can be derived in a fully algorithmic manner using automatic differentiation [123], whereas deriving the continuous adjoint equations is more mathematically involved and often requires domain knowledge for the discretization. Also, accepting a small inaccuracy in the gradients, the memory requirements of the discrete adjoint can be alleviated [30].

In the limit of an infinitely fine grid, given certain conditions required on the primal discretization, both approaches converge to the same value of the gradient [43]. Discretizations with this property are called adjoint or dual consistent [49]. It can therefore be concluded that, in the scope of the optimization problems considered in this paradigm, the choice of adjoint method is mainly implementational [110]. In this work, the continuous adjoint is used to obtain the gradients in the field inversion phase.

3.5. Model problem

In order to make the description of the paradigm more tangible, a one-dimensional model problem from Parish and Duraisamy [107] will be used throughout this work to illustrate the most important steps. This section introduces the problem and describes the steps relevant for the field inversion phase.

3.5.1. Problem description

The true data is generated by the following scalar non-linear ordinary differential equation, representing one-dimensional heat conduction with a radiative and a convective term:

$$\frac{\mathrm{d}^2 T}{\mathrm{d}z^2} = \varepsilon(T)(T_\infty^4 - T^4) + h(T_\infty - T). \tag{3.35}$$

In this expression, $z \in [0,1]$ is the spatial coordinate, the temperature of the surroundings T_∞ can be a function of z, T is the temperature, $h = 0.5$, and

$$\varepsilon(T) = \left(1 + 5\sin\left(\frac{3\pi T}{200} + e^{0.02T} + \mathcal{N}\left(0, 0.1^2\right)\right)\right) \cdot 10^{-4}. \tag{3.36}$$

It is attempted to model this problem using

$$\frac{\mathrm{d}^2 T}{\mathrm{d}z^2} = \varepsilon_0(T_\infty^4 - T^4), \tag{3.37}$$

where $\varepsilon_0 = 5 \cdot 10^{-4}$. This base model is imperfect because it neglects the linear term and approximates the emissivity with a constant. The models are solved using second-order accurate finite volume discretization on a uniform mesh with 31 points. The temperature profiles resulting from the true and the base model are shown in fig. 3.1. An augmented model is obtained by multiplying the base model with the spatially varying corrective term $\beta(z)$:

$$\frac{\mathrm{d}^2 T}{\mathrm{d}z^2} = \beta(z)\varepsilon_0(T_\infty^4 - T^4). \tag{3.38}$$

Using the expressions for the true model and the corrected model, the corrective term corresponding to the true model can be derived to be

$$\beta_{\text{true}} = \beta_r + \beta_c = \frac{1}{\varepsilon_0}\left(1 + 5\sin\left(\frac{3\pi T}{200} + e^{0.02T} + \mathcal{N}\left(0, 0.1^2\right)\right)\right) \cdot 10^{-4} + \frac{h}{\varepsilon_0}\frac{T_\infty - T}{T_\infty^4 - T^4}. \tag{3.39}$$

Coming back to the problem definition for the inversion problem, the goal is to optimize $\beta(z)$ such that when it is used in the corrected model (3.38) the resulting temperature approximates the output of the true model (3.35). Assuming Gaussian distributions for the likelihood and the prior, the objective function is given by

$$J = \frac{1}{2}(\boldsymbol{d} - h(\boldsymbol{\beta}))^T C_m^{-1}(\boldsymbol{d} - h(\boldsymbol{\beta})) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}})^T C_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}}). \tag{3.40}$$

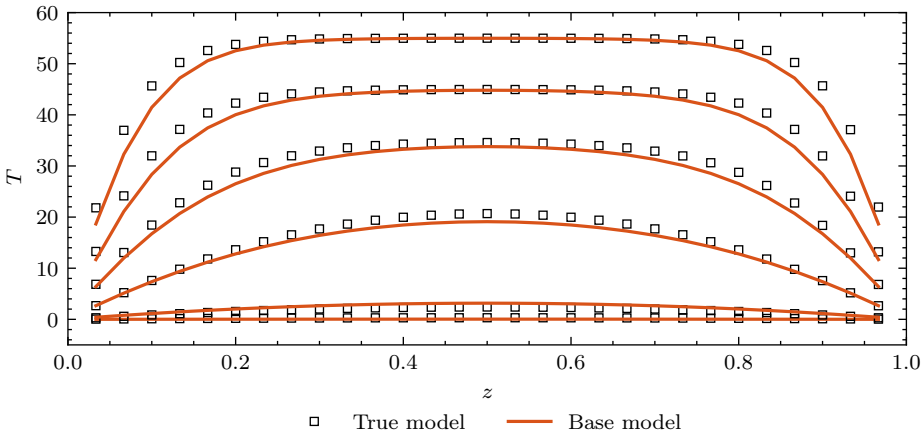


Figure 3.1: Temperature profiles for $T_\infty \in \{5, 15, 25, 35, 45, 55\}$.

The dataset $\mathbf{d}$ consists of 100 realizations of the true model. The observational covariance matrix is constructed according to the three options described in section 3.1. For the diagonal covariance matrix with constant variance, a variance of $\sigma_m^2 = 0.02$ was used. The prior for the corrective term is $\beta_{\text{prior}} = 1$, reducing the augmented model to the base model. Parish and Duraisamy [107] select the constant prior variance by sampling the corrective term from the prior distribution and checking whether the observed temperature profile falls between the resulting $\pm 2\sigma$ bounds. The MAP solution for the particular choice of T_∞ considered in this section used a prior variance of $\sigma_\beta^2 = 0.2$.

3.5.2. Discrete adjoint

The true model can be discretized and solved using the following iterative scheme:

$$T_i = \frac{1}{2}\left(T_{i-1} + T_{i+1} - (\Delta z)^2\left(\varepsilon(T_i)\left(T_i^4 - T_{\infty,i}^4\right) + h(T_i - T_{\infty,i})\right)\right). \tag{3.41}$$

Under-relaxation is used to stabilize the iterations. The base model and augmented model are discretized in a similar manner. From these discretized models, the partial derivatives of the objective function and the governing equations are straightforwardly calculated:

$$\frac{\partial J}{\partial \beta_i} = (C_\beta^{-1})_{ij}(\beta_j - \beta_{\text{prior},j}), \tag{3.42}$$

$$\frac{\partial J}{\partial T_i} = (H_{kl}T_l - d_k)(C_m^{-1})_{jk}H_{ji}, \tag{3.43}$$

$$\frac{\partial R_\alpha}{\partial \beta_i} = \frac{1}{2}(\Delta z)^2\delta_{\alpha i}\varepsilon_0(T_\alpha^4 - T_{\infty,\alpha}^4), \tag{3.44}$$

$$\frac{\partial R_\alpha}{\partial T_i} = \delta_{\alpha i}\left(1 + 2T_\alpha^3(\Delta z)^2\beta_\alpha\varepsilon_0\right) - \frac{1}{2}\delta_{\alpha-1,i} - \frac{1}{2}\delta_{\alpha+1,i}. \tag{3.45}$$

where δ_{ij} is the Kronecker delta, summation is implied over i, j, k, and l, and α is excluded from the summation convention. To obtain the gradient, first use (3.30) to solve for the adjoint variable, i.e.

$$\left(\frac{\partial R_m}{\partial U_k}\right)^T \Psi = -\left(\frac{\partial J}{\partial U_k}\right)^T. \tag{3.46}$$

Then, the resulting adjoint vector Ψ can be used to obtain the gradient using (3.31).

3.5.3. Continuous adjoint

For the continuous adjoint, the derivative of the augmented objective function can be written as

$$\frac{\delta L}{\delta \beta} = \frac{\delta J}{\delta \beta} + \int_0^1 \Psi \frac{\delta R}{\delta \beta}\,\mathrm{d}z. \tag{3.47}$$

Taking the derivative of the primal equation, multiplying by the adjoint variable, and integrating over the domain gives

$$\int_0^1 \Psi \frac{\delta R}{\delta \beta} \mathrm{d}z = \int_0^1 \Psi \frac{\mathrm{d}^2}{\mathrm{d}z^2}\left(\frac{\delta T}{\delta \beta}\right)\mathrm{d}z - \varepsilon_0 \int_0^1 \Psi \left[T^4 - T_\infty^4(z) + 4\beta(z)T^3\frac{\delta T}{\delta \beta}\right]\mathrm{d}z. \tag{3.48}$$

Applying integration by parts twice to the first term on the right-hand side and rearranging the terms gives

$$\begin{aligned}\frac{\delta L}{\delta \beta} = &\underbrace{\left(\Psi \frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\delta T}{\delta \beta}\right) - \frac{\mathrm{d}\Psi}{\mathrm{d}z}\frac{\delta T}{\delta \beta}\bigg|_0^1\right)}_{D^\Psi\left(\frac{\delta U}{\delta \beta}\right)} - \int_0^1 \left(\Psi \varepsilon_0 \left(T^4 - T_\infty^4(z)\right) - \sum_{i=1}^N \left[\delta(z-z_i)\left(\beta - \beta_{\text{prior}}\right)\left(C_\beta^{-1}\right)_{ij}\right]\right)\mathrm{d}z \\ &+ \int_0^1 \underbrace{\left(\frac{\mathrm{d}^2\Psi}{\mathrm{d}z^2} - 4\Psi\varepsilon_0\beta(z)T^3 + \sum_{i=1}^M \left[\delta(z-z_i)(HT(\beta)-d)(C_m^{-1}H)_{ij}\right]\right)}_{R^\Psi}\frac{\delta T}{\delta \beta}\mathrm{d}z.\end{aligned} \tag{3.49}$$

Note that the Dirac delta function has been introduced because the data is known at discrete locations, while the equations are still treated in continuous form. The Dirac delta functions disappear naturally when evaluating the integral in the discretization step. The adjoint boundary conditions follow from setting the first term to zero, and correspond to zero Dirichlet boundary conditions at both boundaries. The adjoint equation follows from setting the indicated part of the last term to zero, using one-point integration for the integral and making use of the sifting property of the Dirac delta function:

$$\left(\frac{1}{\Delta z}\right)\Psi_{i-1} + \left(-2 - 2\varepsilon_0\Delta z\beta_i T_i^3\right)\Psi_i + \left(\frac{1}{\Delta z}\right)\Psi_{i+1} = (HT(\beta)-d)C_m^{-1}H. \tag{3.50}$$

The remaining term provides the adjoint gradient. Again, making use of the sifting property of the Dirac delta function and using one-point integration for the integral, the gradient can be rewritten as

$$\frac{\delta L}{\delta \beta} = -\Psi_i \varepsilon_0 \left(T_i^4 - {T_{\infty,i}}^4(z)\right)\Delta z - \left(\beta - \beta_{\text{prior}}\right)C_\beta^{-1}. \tag{3.51}$$

3.5.4. Hessian approximation

As discussed in section 3.1, the Gauss-Newton approximation can be used to approximate the Hessian, which can be used to approximate the posterior covariance matrix. In order to investigate the impact of this assumption, it is also used in this section to approximate the posterior covariance. Using diagonal observational and prior covariance matrices with constant variance, the expression for the approximate Hessian can be derived from (3.16) as:

$$\frac{\delta^2 J}{\delta\beta_j\delta\beta_k} \approx \frac{1}{\sigma_m^2}\sum_{i=1}^N \frac{\delta T_i}{\delta\beta_j}\frac{\delta T_i}{\delta\beta_k} + \frac{1}{\sigma_\beta^2}\delta_{jk}. \tag{3.52}$$

The first term can be obtained by calculating the adjoint gradient of the following objective function:

$$J_i = \frac{T_i - T_{\text{obs},i}}{\sigma_m}, \tag{3.53}$$

which changes the partial derivatives of the objective function as follows:

$$\frac{\partial J_i}{\partial \beta_j} = 0 \quad \text{and} \quad \frac{\partial J_i}{\partial T_j} = \frac{1}{\sigma_m}\delta_{ij}. \tag{3.54}$$

The Hessian can then be approximated by solving the adjoint equations with this objective function for each $i \in [1, N]$.

3.5.5. Results

A convenient way to verify the implementation of the adjoint equations before performing the optimization is by using both the adjoint method and finite differences to obtain the gradients. The gradients for the model problem at $\beta(z) = 1$ obtained using finite differences, the discrete adjoint, and the continuous adjoint, are

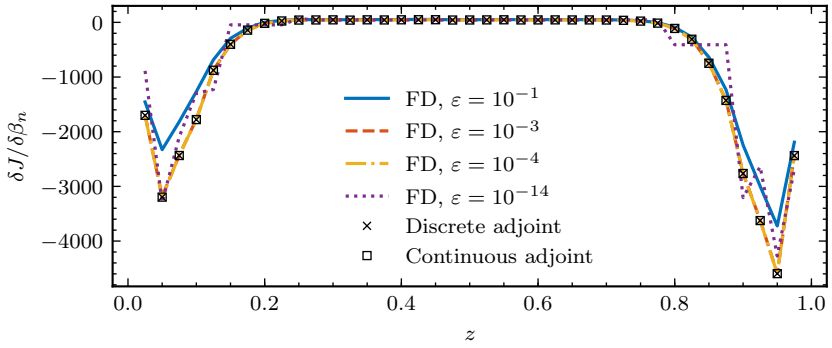


Figure 3.2: Verification of the gradient calculation.

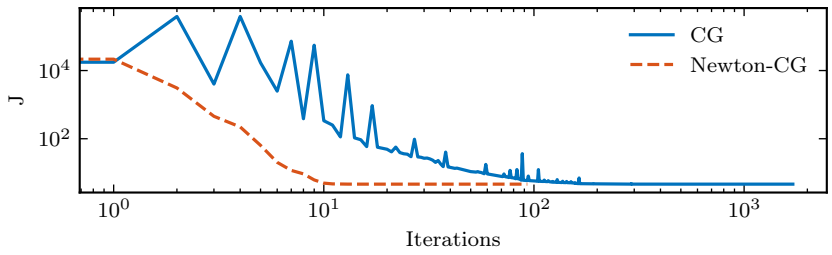


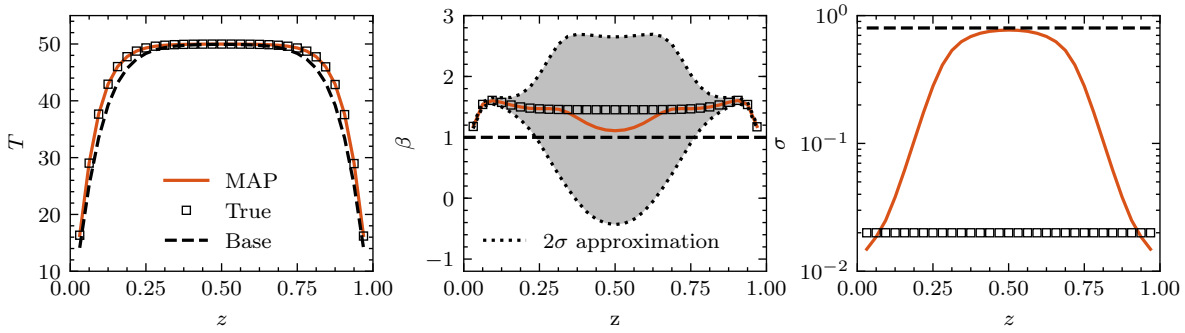
Figure 3.3: Objective function during optimization using first- and second-order optimization algorithms.

presented in fig. 3.2. As expected from the discussion in section 3.3.3, choosing an excessively low or high perturbation size for the finite difference calculation results in inaccuracies in the gradient. For intermediate perturbation sizes, the gradients obtained from finite differences and the discrete and continuous adjoint are practically indistinguishable.

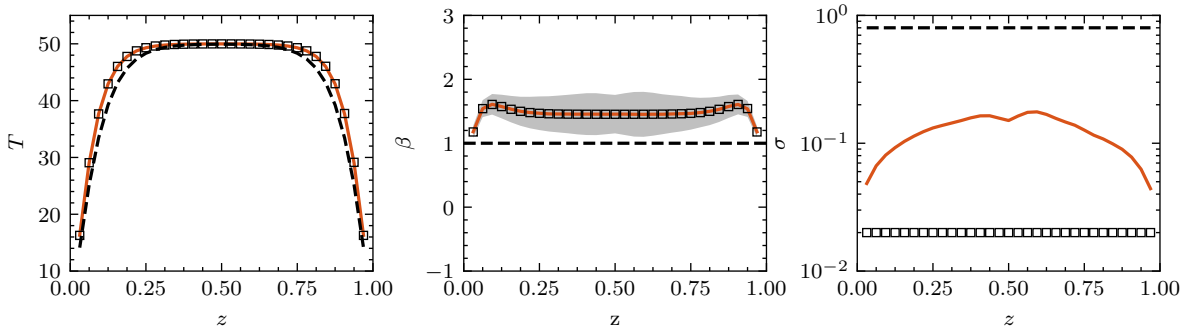
The gradients can now be used in an optimization routine to calculate the MAP of the corrective term. The convergence of the optimization is shown for two optimization methods in fig. 3.3. As discussed in section 3.3.1, conjugate gradients (CG) only uses gradient information, whereas the Newton-CG algorithm additionally uses the Hessian. In this case, the Hessian is calculated using the discrete direct-adjoint method. The derivation can be found in appendix A. Both methods reduce the objective function by several orders of magnitude and converge to the same solution. It can be seen that CG converges two orders of iterations later than Newton-CG. It should be noted that each iteration consists of multiple system solves. However, even taking into account the system solves needed for the gradient and Hessian evaluations, CG converges in approximately 2000 function calls, whereas Newton-CG only takes around 600.

The results of the field inversion, using the three options for the covariance matrix mentioned in section 3.1, are shown in fig. 3.4. The resulting temperature profile agrees with the true solution for all covariance matrices. The same is true for the corrective term, except for the objective function where the diagonal covariance matrix with constant variance was used. The standard deviation is only accurate when the full observational covariance matrix is used. For the diagonal covariance matrix with constant variance, there is a strong agreement between the standard deviation obtained using the Gauss-Newton approximation of the Hessian with that calculated using the exact Hessian. This finding is of course not guaranteed to generalize to more complex problems.

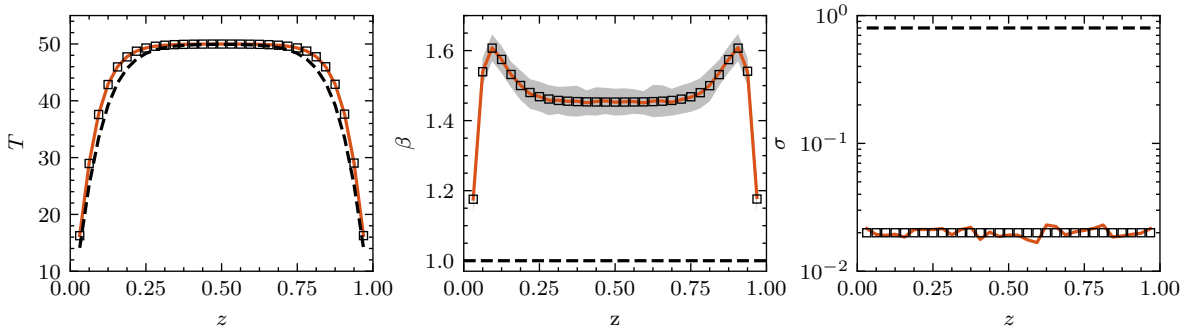
It can thus be concluded that an accurate description of the observational covariance is important for the inferred corrective term to be physically meaningful, whereas the temperature distribution can be inferred accurately as long as the confidence in the observations is high enough (i.e. the observational variance is low enough). This is also true for the approximation of the posterior variance, which was only accurate for the most detailed description of the observational covariance. The MAP solutions for all values of T_∞ in the dataset have been separated into the radiative and the convective term as defined in (3.39), and is shown in fig. 3.5 for the three different covariance matrices. Uncertainty bounds are shown for the radiative term. Both parts of the corrective term have been inferred correctly, especially for the full covariance matrix.



(a) Diagonal covariance matrix with constant variance, $\sigma_m^2 = 0.02$.

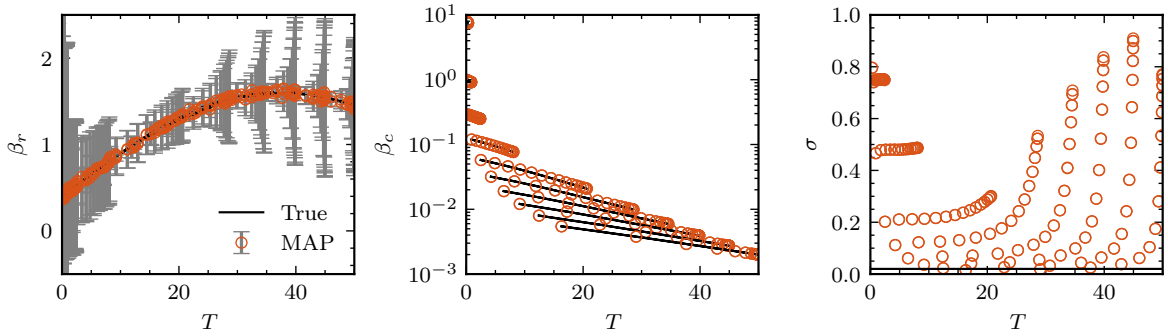


(b) Diagonal covariance matrix with vector variance.

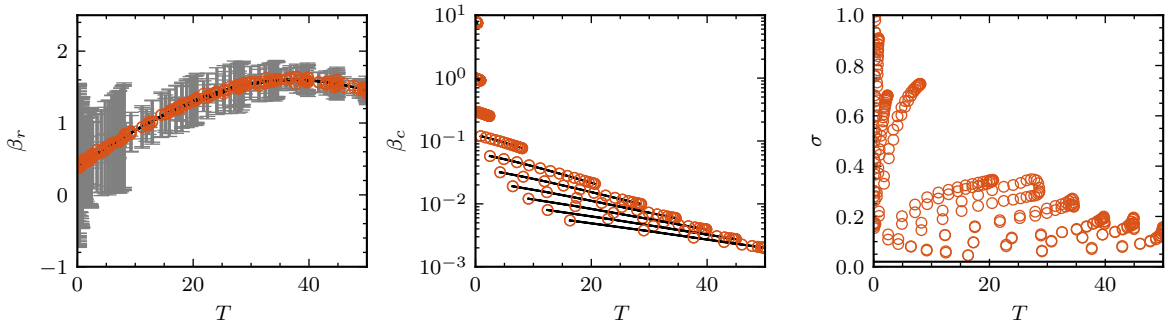


(c) Full covariance matrix.

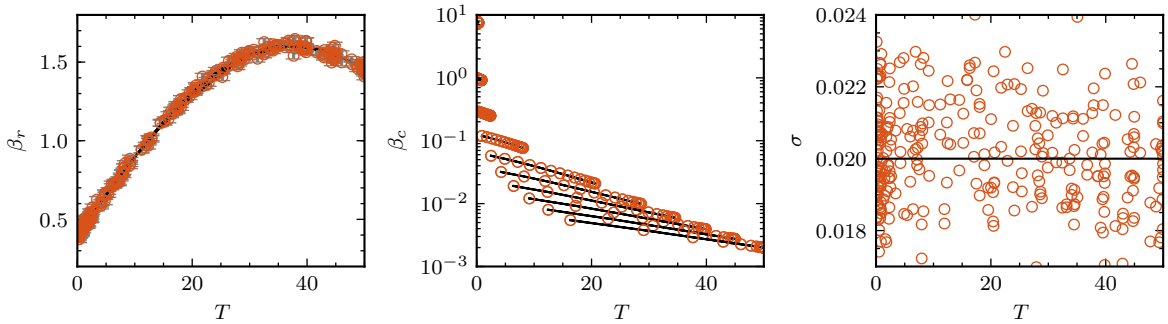
Figure 3.4: Field inversion results ($\pm 2\sigma$ for the corrective term), $T_\infty = 50$, $\sigma_\beta^2 = 0.8$.



(a) Diagonal covariance matrix with constant variance, $\sigma_m^2 = 0.02$.



(b) Diagonal covariance matrix with vector variance.



(c) Full covariance matrix

Figure 3.5: Inferred radiative and convective corrective terms for all T_∞, $\sigma_\beta^2 = 0.8$.

4

Machine learning

This chapter will provide the theoretical basis for the second part of the paradigm - machine learning. In section 4.1, a general description of the types of problems addressed by machine learning is given, and the most important challenges and trade-offs are discussed. Section 4.2 gives an overview of the machine learning algorithms which are relevant in the context of data-driven turbulence modeling. For each algorithm, the major strengths, shortcomings, and capabilities for its use in a probabilistic setting are discussed. Furthermore, examples of their use in the field of turbulence modeling are given. The input features are discussed in section 4.3 and section 4.4 describes the process of model selection. Finally, in section 4.5 some of the points are illustrated for the simple model problem introduced in the previous chapter.

4.1. Problem formulation

Machine learning can be defined as a method to discover patterns in data, which can then be used to make predictions on unseen data [94]. A frequently made distinction is between supervised and unsupervised learning. In the first, a mapping between inputs $\boldsymbol{x}$ and outputs y is approximated, given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$. For classification tasks, the output consists of a set of classes $y \in \{1, \dots, C\}$. In regression problems, the output is real-valued. In the unsupervised case, the dataset only consists of a set of inputs $\mathcal{D} = \{(\boldsymbol{x}_i\}_{i=1}^{N}$. The goal is then to discover patterns in this dataset. Besides supervised and unsupervised learning, reinforcement learning is a branch of machine learning which aims to find an optimal policy which translates observed states into actions given an occasional award or punishment [143].

In practice, many machine learning algorithms can be made as complex as desired. However, models with high complexity have the risk of learning non-existing patterns in noise in the training data. The model is then said to have low bias but high variance, a phenomenon called overfitting. On the other hand, underfitting occurs when models with insufficient complexity fail to fit even the basic trend in the data. The model is then said to have high bias but low variance. This trade-off is therefore often referred to as the bias-variance trade-off, and is an essential challenge in the development and use of machine learning algorithms.

Another important problem in machine learning is the curse of dimensionality. This problem arises in high-dimensional spaces, i.e. in cases where a large number of features is used. With increasing dimensionality, the volume of the feature space increases significantly, causing the data to sparsify. This causes problems in methods using distance measures (e.g. k-nearest neighbours (k-NN), support vector machines (SVM), etc.), significantly increases the number of samples needed to obtain a certain sampling density, and poses challenges in optimization problems.

4.2. Overview of machine learning algorithms

This section gives a brief overview of the machine learning algorithms relevant to this work. The discussion is mainly based on the theory presented in Murphy [94], unless stated otherwise.

4.2.1. Kernel regression

Kernel regression is a simple algorithm which aims to compute the conditional expectation

$$f(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}] = \int y p(y|\boldsymbol{x}) dy = \frac{\int y p(\boldsymbol{x}, y)\,\mathrm{d}y}{\int p(\boldsymbol{x}, y)\,\mathrm{d}y}, \tag{4.1}$$

using a kernel density estimate for the joint probability, i.e.

$$p(\boldsymbol{x}, y) \approx \frac{1}{N}\sum_{i=1}^{N} \kappa_h(\boldsymbol{x} - \boldsymbol{x}_i)\kappa_h(y - y_i). \tag{4.2}$$

From the zero-mean property of the kernel and the fact that it integrates to one, it easily follows that

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} w_i(\boldsymbol{x})\, y_i \qquad \text{and} \qquad w_i(\boldsymbol{x}) \equiv \frac{\kappa_h(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{j=1}^{N} \kappa_h(\boldsymbol{x} - \boldsymbol{x}_j)}. \tag{4.3}$$

An advantage of kernel regression is that it is non-parameteric. Therefore, training the algorithm is restricted to tuning the hyperparameters of the kernel. On the other hand, kernel regression has a relatively high computational complexity in terms of memory demands and evaluation time. In the context of data-driven turbulence modeling, kernel regression has been used by Tracey et al. [151] to construct a probabilistic model of the error of RANS models using high-fidelity data.

4.2.2. Neural networks

Feedforward neural networks or multilayer perceptrons consist of a network of units which transform a linear combination of their inputs through a non-linear activation function [45]. The network is usually made up of an input layer, followed by a number of hidden layers resulting in an output layer. Each layer is made up of a number of perceptrons. A perceptron takes the weighted input of the previous layer, adds a bias, and applies an activation function. An example of a perceptron is shown in fig. 4.1. Note that the weight w_1 multiplying the unit input is equivalent to adding a bias term. The gradient of the loss function with respect to the weights and biases are calculated using the backpropagation algorithm. This algorithm consists of a forward-phase, in which the data is propagated through the network and the outputs are calculated. The discrepancy between these outputs and the training labels is quantified using a loss function. The gradients are then calculated by repeatedly applying the chain rule, working backwards from the output layer.

Learning the weights of a neural network can be formulated as a maximum-likelihood estimation:

$$\boldsymbol{w}^{\mathrm{MLE}} = \arg\max_{\boldsymbol{w}} p(\mathcal{D}|\boldsymbol{w}). \tag{4.4}$$

Neural networks can be used in a probabilistic setting using a Bayesian approach. In this case, one can define probability distributions on the weights, and then find the maximum a posteriori given a prior on the weights:

$$\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} \left\{\log p(\mathcal{D}|\boldsymbol{w}) + \log p(\boldsymbol{w})\right\}. \tag{4.5}$$

A normally distributed prior on each weight corresponds to L2-regularization, whereas a Laplace prior corresponds to L1-regularization [46]. Due to the high number of weights in most neural network architectures, calculating the posterior is intractable. Blundell et al. [12] assign a normal distribution to each weight of the neural network, and use variational inference to infer the individual means and standard deviations. However, it is often assumed that there is no correlation between the weights. Furthermore, the quality of the resulting uncertainty bounds can depend considerably on the architecture of the network. Advantages of neural networks are their flexibility and their ability to learn complex non-linear functions. On the other hand, the training time can be rather long and a relatively large amount of training data is required. Also, because of its flexibility there are a lot of hyperparameters to tune and the results are hard to interpret.

A large part of the work done so far on data-driven modeling uses neural networks [29, 81, 138, 152, 171]. Most of these implementations use simple feedforward neural networks, but [82] proposes an architecture with a layer with an invariant tensor basis to ensure Galilean invariance of the resulting turbulence model.

4.2.3. Support vector machines

Support vector machines (SVM) are a class of machine learning algorithms which aim to separate classes of training data using a hyperplane described by

$$\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = 0, \tag{4.6}$$

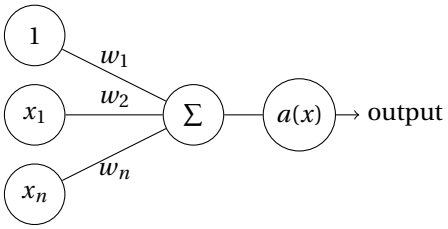


Figure 4.1: Diagram of a perceptron.

while maximimizing the margin between the hyperplane and the nearest datapoints [23]. Requiring that each point in the dataset must lie on the correct side of the hyperplane, this can be written as the following optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{w},\boldsymbol{b}}{\text{minimize}} \quad ||\boldsymbol{w}|| \\ &\text{subject to} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i - \boldsymbol{b}) \geq 1,\ i = 1,\ldots,n. \end{aligned} \tag{4.7}$$

This is the so-called *hard-margin* SVM. In practice, there usually does not exist a hyperplane which can separate all data points. In this case, the data is not linearly separable. The soft-margin formulation allows some datapoints to be misclassified. Furthermore, the input data can be efficiently mapped to a high-dimensional space using kernels, allowing for learning non-linear decision boundaries. Another advantage is that only the data points which directly influence the decision boundary, the support vectors, are used in the decision process. This allows for efficient memory utilization. Support vector machines are mainly used for binary classification, however, extensions to multiclass classification [54] and regression [27] are possible.

Support vector machines have not seen frequent use in data-driven turbulence modeling. An example is the work of Ling and Templeton [80], where they are used to identify regions of high RANS uncertainty. However, it was found that random forests performed better in terms of performance and ease of implementation, as support vector machines are more sensitive to irrelevant input features.

4.2.4. Gaussian processes

The goal of Gaussian processes is to infer a distribution over functions, given the data [125]. The main idea is that datapoints which are close together in the feature space should have similar target values. This closeness is quantified using a kernel function. Assume that the observed labels can be described by

$$y = f(\mathbf{x}) + \varepsilon, \qquad \text{where} \qquad \varepsilon \sim \mathcal{N}(0, \sigma_y^2), \tag{4.8}$$

and where $f(\boldsymbol{x})$ is a function of the input data and σ_y^2 is the variance of the noise. In this work, σ_y^2 can, for example, be chosen as the variances obtained from the MAP result. The covariance matrix of the labels is given as

$$\text{cov}[\mathbf{y}|\mathbf{X}] = \mathbf{K} + \sigma_y^2\mathbf{I} \equiv \mathbf{K}_y, \tag{4.9}$$

where $\mathbf{K} = \kappa(\mathbf{X},\mathbf{X})$ with training data $\mathbf{X}$ and a positive definite kernel κ. The test set will be written as $\mathbf{X}^*$. The posterior is then given by the multivariate Gaussian

$$p(\mathbf{f}_*|\mathbf{X}_*,\mathbf{X},\mathbf{y}) = \mathcal{N}(\boldsymbol{f}_*|\boldsymbol{\mu}_*,\boldsymbol{\Sigma}_*), \tag{4.10}$$

where

$$\boldsymbol{\mu}_* = \boldsymbol{K}_*^T\boldsymbol{K}_y^{-1}\boldsymbol{y}, \qquad \boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T\boldsymbol{K}_y^{-1}\boldsymbol{K}_*, \tag{4.11}$$

and $\mathbf{K}_* = \kappa(\mathbf{X}_*,\mathbf{X})$, and $\mathbf{K}_{**} = \kappa(\mathbf{X}_*,\mathbf{X}_*)$. A common choice for the kernel is the radial basis function (RBF) kernel, given by

$$\kappa(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||}{l^2}\right). \tag{4.12}$$

In this case, the only hyperparameter to be tuned is the length scale l. This can be done by optimizing the log marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X}) = \log\mathcal{N}(\boldsymbol{y}|\boldsymbol{0},\boldsymbol{K}_y) = -\frac{1}{2}\boldsymbol{y}\boldsymbol{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_y| - \frac{N}{2}\log(2\pi). \tag{4.13}$$

The first term in this expression expresses the misfit between the outcomes and the data, the second term penalizes an overly complex model, and the third term is a normalization constant. The gradients can be readily computed from this expression, and it is thus convenient to use a gradient-based optimization algorithm to tune the hyperparameters. However, it is also possible to use sampling methods such as MCMC for this purpose.

Examples of Gaussian processes in data-driven turbulence modeling are the works by Parish and Duraisamy [107] and Zhang and Duraisamy [171]. Gaussian processes are easy to implement and are inherently probabilistic. However, the naive implementation has a time complexity of $O(N^3)$, where N is the number of training samples. Quinonero-Candela et al. [121] give an overview of efficient approximations for Gaussian process regression. Zhang and Duraisamy [171] and Zhang et al. [170] investigate multiscale Gaussian process regression.

4.2.5. Tree-based methods

The main idea of tree-based methods is to segment the feature space into M regions $R_1, R_2, \ldots, R_M$ using trained splitting rules [38]. A tree consists of several internal nodes connected by branches ending in a set of terminal nodes, and is built by recursively finding the split of the input space which minimizes a given loss function. Prediction samples falling into a given terminal node are given a constant value c_m. The function describing the decision tree is then given by

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m), \tag{4.14}$$

where I is a function indicating whether x falls in region R_m. For example, c_m can be chosen as the average of all samples ending up in a given terminal node in a regression problem. This corresponds to finding the minimum sum of squares. Tree-based methods can be applied to both regression and classification problems. Bias and variance can be traded off by varying the tree size, where larger trees have more tendency to overfit. Advantages of using trees to are that they are easy to interpret, can handle a mix of discrete and continuous features, and usually scale to large datasets [94].

A tree-based method therefore needs to find which features to use and where to apply the split at each node in the tree. Furthermore, it needs to find the topology of the tree. Minimizing the sum of squares for all combinations of features and splits generally has an excessively high computational cost. Therefore, a greedy algorithm is used [38]. At a given node, the split at s in the feature space for feature j can be written as

$$R_L(j,s) = \{\boldsymbol{X} | [\boldsymbol{X}_j] \leq s\} \qquad \text{and} \qquad R_R(j,s) = \{\boldsymbol{X} | [\boldsymbol{X}_j] > s\}. \tag{4.15}$$

The splitting feature j and splitting point s are then given by the following optimization problem:

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_L(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_R(j,s)} (y_i - c_2)^2 \right]. \tag{4.16}$$

Generally, single trees perform significantly worse than other machine learning algorithms [38]. This is partly cause by the unstable nature of trees: a small change in the input can change the tree completely. Trees are thus generally high-variance estimators. However, individual trees can be combined into an ensemble to reduce the variance and thus to improve performance. One option is to take repeated samples from the dataset (i.e. bootstrapping) and training a model on each of the bootstrapped datasets. The prediction is then obtained by aggregating the results of the individual models, for example by taking the mean for regression or using a majority vote in the case of classification. Combining these models reduces the variance of the prediction, given that the individual models are sufficiently different from each other. Building on this idea, random forests aim to increase the diversity of the models by only allowing to consider $m < p$ features for each split, where p is the total number of features. This decorrelates the trees, as it discourages choosing a small number of good predictors for the majority of the splits, thus allowing for greater variance reduction. A third option is to improve trees in specific areas where the model is not performing well, by growing the trees sequentially and fitting each model to the residuals of the previous model. This approach is called boosting.

Random forests are used by Ling and Templeton [80], Ling et al. [81], and Wang et al. [157]. A boosting algorithm called Adaboost is used by Ling and Templeton [80]. Recent work has applied the concept of using a tensor basis for neural networks [82] to random forests [67, 68].

4.2.6. Other algorithms

There are several algorithms which are not discussed in detail in this section but which do appear in literature. One example is the work by the group of Weatheritt and Sandberg [159, 160], who use symbolic regression and gene expression programming to obtain tangible analytical expressions for the Reynolds stress anisotropy tensor. Another example is the study by Ray et al. [126], who formulate a nonlinear eddy viscosity model and uses shrinkage regression to remove most of its terms, keeping only the most important ones. Then, a Bayesian calibration using MCMC is performed to tune the model parameters. Sekar et al. [132] use convolutional neural networks to obtain an airfoil shape given a pressure distribution.

4.3. Features

4.3.1. Feature engineering

The features to be used as input in the machine learning algorithm must be informative enough to distinguish individual points in the feature space. Also, it must be possible to derive them from mean flow quantities, as they have to be available during the RANS simulation. Furthermore, in order to aid generalization of the predictions, the features should be Galilean invariant and should not depend on the geometry. For some machine learning algorithms, the features should also be normalized. From the first applications of machine learning in turbulence modeling to the more recent studies, the features moved from ad-hoc selected, variant input features trained and tested on the same flows to systematically selected, invariant input features tested on multiple flows.

Milano and Koumoutsakos [92] used the instantaneous velocity as input to predict the velocity in the near-wall region. The method was trained and tested on the same flow configuration, thus it was not necessary to use rotational invariant features. A more systematic approach to feature selection was used by Duraisamy et al. [29], where the hill-climbing method proposed by Kohavi and John [72] was used to select the input features from a feature set consisting of the velocity gradient tensor, the transported turbulent scalar quantities, and three non-dimensional parameters. A first attempt to take into account rotational invariance was taken by Lefik and Schrefler [79], who trained their algorithm on multiple rotations of the original dataset to try to let the algorithm learn the invariance itself. This is a common technique in the domain of image recognition [6]. Several points can be raised arguing that this approach to incorporating invariance in the system is sub-optimal [81]. Firstly, the rotational invariance learned by the algorithm is not exact. In physical models exact rotational invariance is desired, whereas in image recognition this can actually be undesirable. Secondly, in physical models it is often possible to derive invariant input features. Not having to train the model on a large number of variations of the original training data can thus save computational time in the training phase.

Invariant input features have been used in several studies [80, 151, 157]. However, the features were selected based on physical reasoning and intuition. The risk of this approach is that important information is excluded by not considering one or more invariants. This was improved upon in the work of Ling et al. [81], by using a systematic approach to derive invariant bases to be used as input features.

4.3.2. Feature importance

One of the main strengths of decision trees is their high interpretability. The decision rules used to obtain the regression value can be visualized in two-dimensional structure [38]. This advantage is lost when an ensemble of decision trees is used. However, it is generally possible to obtain an estimate of the relative importance of each of the features in the dataset. As discussed in section 4.2, the feature space is split according to the maximal improvement of fit compared to assigning a constant value over the full region. These improvements can be summed for each feature to indicate its relative importance. As the absolute values do not have a meaning, the importances are usually scaled with the highest importance or such that all importances sum to one. The squared relative importance for feature j is given by

$$\mathcal{I}_j^2 = \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M}\left[\mathrm{MSE}_{R_L+R_R} - \frac{1}{n_{s,R_L}+n_{s,R_R}}\left(\mathrm{MSE}_{R_L}\, n_{s,R_L} + \mathrm{MSE}_{R_R}\, n_{s,R_R}\right)\right], \tag{4.17}$$

where T is the number of trees and M is the number of internal nodes where feature j was used to make the split.

T_∞	σ_{prior}
5	20
10	2
15	1
20	1
25	0.5
30	1
35	1
40	1
45	1
50	0.8

Table 4.1: Settings used for the field inversions making up the dataset, taken from Parish and Duraisamy [107].

4.4. Model selection

In order to know how well a statistical method is performing or to tune its hyperparameters, an estimate of the error on unseen data is needed. There are several approaches, differing in the computational cost and the bias and variance of the estimate of the test error [38]. In the *validation set approach*, the model can be trained on some portion of the complete dataset and tested on the remaining samples. This approach is conceptually simple, but it generally overestimates the test error because the model is not trained using all the available data. Furthermore, the estimate of the test error can vary significantly depending on which samples are included in the test set. In *leave-one-out cross-validation* (LOOCV) only one sample is left out of the dataset to estimate the test error, and the algorithm is trained on the remaining $n-1$ samples. This process is repeated n times, until each sample in the dataset has been used. With this method, almost the complete dataset is used for the training, and thus the estimate of the test error is almost unbiased. However, because the various training sets are similar they are highly correlated, and the estimate of the test error has high variance. Furthermore, if the dataset is large or if the training time of the algorithm is high, this approach can be computationally expensive. The bias and variance can be traded-off using k-fold cross-validation. In this case, the dataset is split up into k approximately equally sized parts, and each part is excluded to estimate the test error once. It has been empirically found that using $k = 5$ or $k = 10$ provides a good balance between bias and variance [14].

4.5. Model problem

In the machine learning phase for this model problem, the machine learning features are T_∞ and the resulting temperature distribution T from the base model. The goal is to approximate a function between these features and the MAP result of a number of field inversions. The dataset used for the machine learning phase thus consists of the inferred corrective terms for ten (constant) values of T_∞. For each case, a prior of $\beta_{\text{prior}} = 1$ is used with a constant diagonal covariance matrix with variances as given in table 4.1. For all cases a full covariance matrix was used. For this machine learning phase, Gaussian processes were used. The noise added to the kernel matrix, σ_y as in (4.8), was taken from the posterior covariance resulting from the field inversion phase. Note that not the full posterior covariance matrix was used, but only the variances. The lengthscale of the RBF kernel was optimized offline, resulting in $l = 4.17$. The resulting corrective function was propagated once through the augmented forward model.

The Gaussian process can then be used to predict the corrective function for an unseen T_∞. As a test case, $T_\infty(z) = 15 + 5\cos(\pi z)$ was selected. This choice differs from the training data in the sense that T_∞ now varies with z, whereas it was constant for all training cases. Furthermore, the low values result in a relatively high influence of the linear heat transfer term [107], which is missed by the base model. The predicted corrective term is compared to the base model and the MAP result in fig. 4.2. The predicted corrective function roughly agrees for $z < 0.5$, but varies considerably from the MAP result for the other end of the rod. Interestingly, the machine learning result is closer to the true corrective term than the MAP solution near the ends of the rod.

The resulting temperature distribution for $T_\infty(z) = 15 + 5\cos(\pi z)$ is shown in fig. 4.3. The temperature resulting from the augmented model is much closer to the true model than the temperature distribution from the base model. Also, the posterior covariance accurately reflects the discrepancy between the results from the true model and the augmented model, i.e. the standard deviation is higher where the discrepancy is higher, and vice versa. At the endpoints of the domain, where the difference is zero, the posterior variance

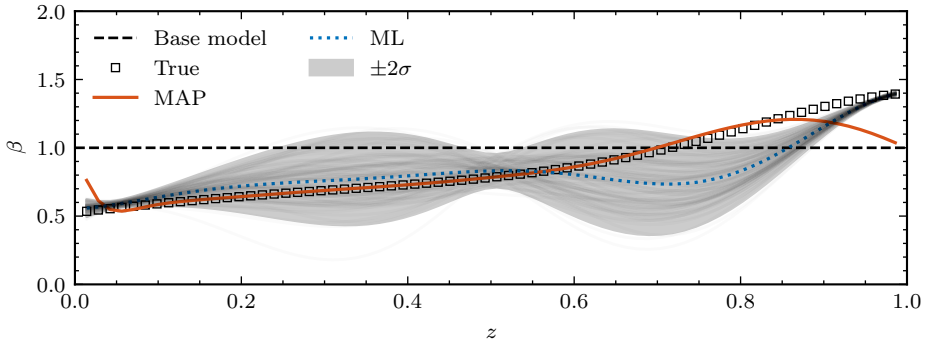


Figure 4.2: Predicted corrective function and samples of the Gaussian process.

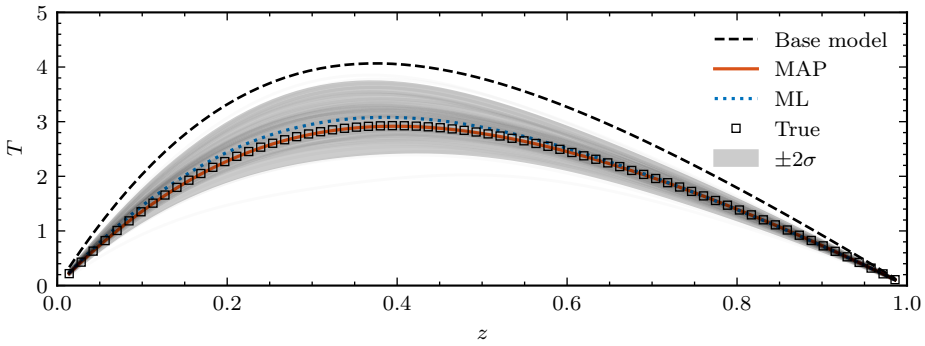


Figure 4.3: Resulting temperature profile $T \pm 2\sigma$ of propagated machine learning predictions, $T_\infty = 15 + 5\cos(\pi z)$.

goes to zero.

The relation between the standard deviation given by the propagated predictions of the corrective term and the discrepancy between the predicted mean temperature and the true temperature is given in fig. 4.4. There is a clear correlation between the error in the predicted temperature and the predicted variance. This shows that, for this case, the variance from the propagated samples is a good indicator of the discrepancy in the results. Similar correlation was observed for other test cases and is also reported by Parish and Duraisamy [107].

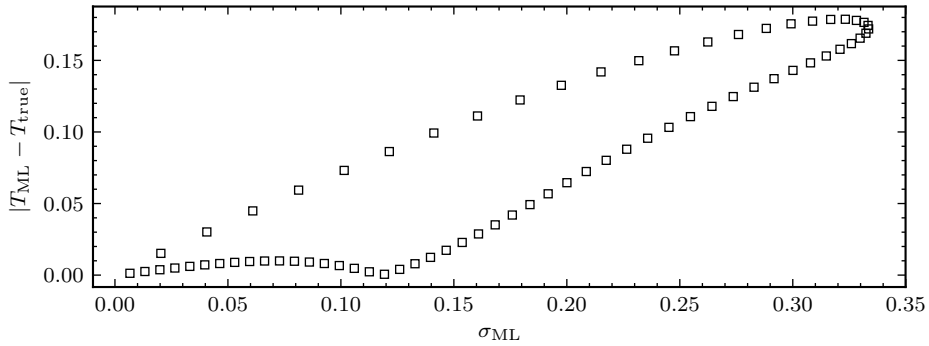


Figure 4.4: Relation between the temperature error and the standard deviation from the propagated Gaussian process samples, $T_\infty = 15 + 5\cos(\pi z)$.

5

Flow cases

This chapter introduces the five flow cases considered in this work, and discusses the sources which were used for the high-fidelity data. For each case, the flow characteristics, quantities of interest, and boundary conditions are described. An overview of the cases, their Reynolds numbers, and their abbreviations are given in table 5.1. Furthermore, the results of the mesh convergence studies are shown for each case.

Case	Abbreviation	Reynolds numbers
Turbulent channel flow	TCF	180, 395, 550, 590, 950, 2,000, 4,200
Square duct	SD	1,100, 1,800, 2,400, 2,900, 3,500
Periodic hills	PH	5,600, 10,595
Converging-diverging channel	CDC	12,600
Backward facing step	BFS	5,100

Table 5.1: Overview of flow cases.

5.1. Turbulent channel flow

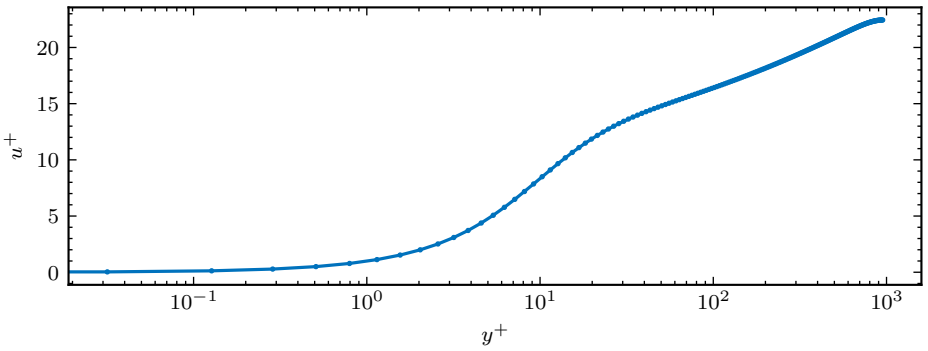


Figure 5.1: Mean velocity profile of the turbulent channel flow case ($\mathrm{Re}_\tau = 950$).

The simplest flow case considered in this work is the flow between two parallel plates, both with infinite dimensions in the streamwise and spanwise directions. The bottom and the top wall are located at $y = 0$ and $y = 2\delta$, respectively, where δ is the channel half-height. Furthermore, it is assumed that the flow is fully developed, i.e. the velocity statistics are independent of x. The Reynolds number is based on the channel half-height and the friction velocity, defined by

$$u_\tau \equiv \sqrt{\frac{\tau_w}{\rho}}, \tag{5.1}$$

where the density $\rho = 1$ throughout the domain and τ_w is the wall shear stress. Cyclic boundary conditions are applied in the streamwise and spanwise directions and the flow is forced by a constant forcing term given by

$$\frac{\mathrm{d}p_w}{\mathrm{d}x} = -\frac{u_\tau^2}{h}, \tag{5.2}$$

which can be easily derived from the properties of the flow domain [116, p. 266]. The results are often expressed in wall units, given by $y^+ = yu_\tau/\nu$ and $u^+ = u/u_\tau$. Furthermore, a no-slip boundary condition is applied to the bottom wall and a symmetric boundary condition is applied to the midplane, such that only one half of the domain is simulated. For all Reynolds numbers, the first grid node was place well into the viscous sublayer. For the Reynolds numbers $\mathrm{Re}_\tau = 180$, 395, and 390, the data was obtained from Moser et al. [93], the cases $\mathrm{Re}_\tau = 550$, 950, 2,000 from Hoyas and Jiménez [53], and the highest Reynolds number case in which $\mathrm{Re}_\tau = 4{,}200$ from the study performed by Lozano-Durán and Jiménez [83].

5.2. Square duct

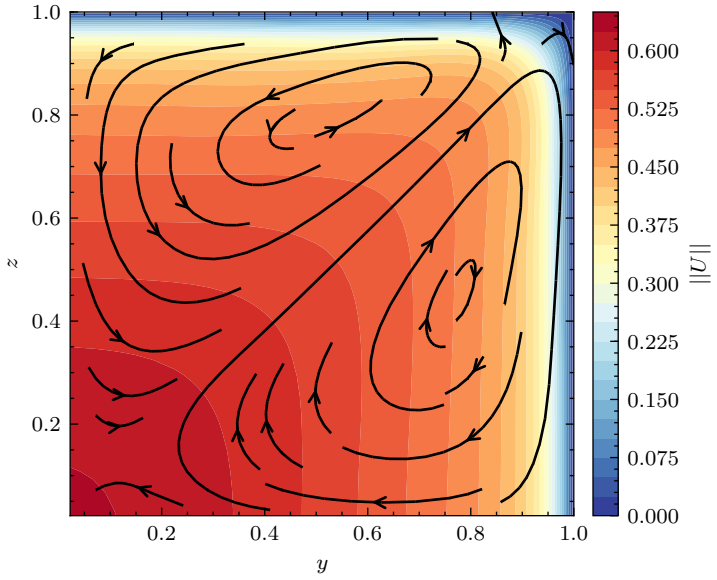


Figure 5.2: Mean velocity magnitude, square duct (Re = 3,500).

The simplest two-dimensional flow case considered in this study is the flow through a square duct. The data for the square duct flow case is obtained from Pinelli et al. [111]. The Reynolds number ranges from 1,100 to 3,500, and is based on the bulk velocity and the semi-height of the duct. As this flow case is symmetric, only one quarter of the flow domain is considered in the computations. An important feature of this flow case is the so-called secondary flow motions. These motions are interesting because linear eddy viscosity models are not able to calculate them. This can be easily shown by considering the source term in the streamwise component of the Reynolds averaged vorticity equation:

$$\begin{aligned} \varepsilon_{1jk}\frac{\partial^2 R_{km}}{\partial x_j \partial x_m} &= \varepsilon_{123}\left(\frac{\partial^2 R_{32}}{\partial x_2 \partial x_2} + \frac{\partial^2 R_{33}}{\partial x_2 \partial x_3}\right) + \varepsilon_{132}\left(\frac{\partial^2 R_{22}}{\partial x_3 \partial x_2} + \frac{\partial^2 R_{23}}{\partial x_3 \partial x_3}\right) \\ &= \left(\frac{\partial^2}{\partial x_2 \partial x_2} - \frac{\partial^2}{\partial x_3 \partial x_3}\right) R_{23} - \frac{\partial^2}{\partial x_2 \partial x_3}\left(R_{22} - R_{33}\right) = 0, \end{aligned} \tag{5.3}$$

making use of the symmetry of the Reynolds stress tensor, the fact that the flow is fully developed, and the fact that $R_{22} = R_{33}$. Flows of engineering interest where secondary flow motions are important are for example wing-body or hub-blade junctions. LEVMs predict too early separation in these geometries in the presence of an adverse pressure gradient, partly because of the absence of secondary motions [91].

On the walls, no-slip boundary conditions are imposed, whereas symmetric boundary conditions are used on the centerlines. Furthermore, a source term is added to the momentum equation, as the flow is

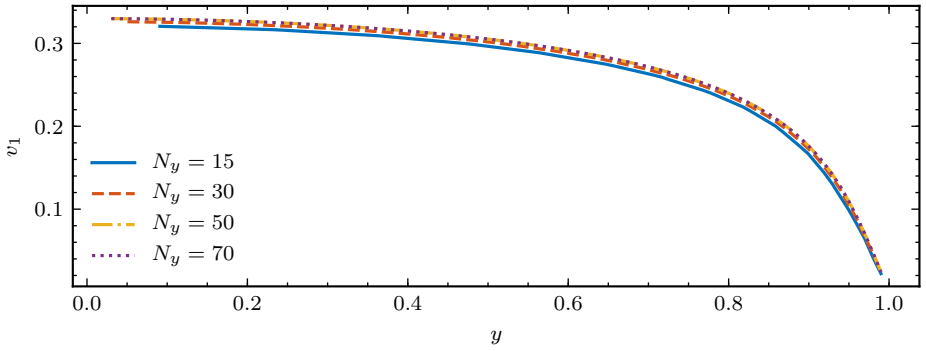


Figure 5.3: Streamwise velocity in square duct along $z = 0.95$ for various mesh refinements (Re = 3,500).

periodic. The mesh contains 40 cells in each direction for the case where Re = 1100 and 60 cells for the case where Re = 3500. For all meshes, the value of y^+ was calculated using the OpenFOAM utility, and was kept below a value of 1 along the walls. A grid convergence study was performed, of which the results are shown in fig. 5.3 for Re = 3500. The figure shows the streamwise velocity along $z = 0.95$. The difference between the streamwise velocities of the two finest grids is negligible.

5.3. Periodic hills

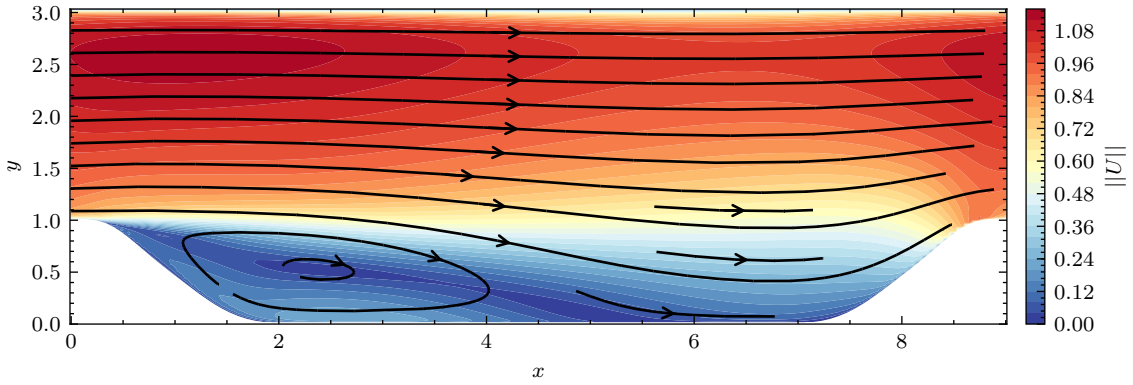


Figure 5.4: Mean velocity magnitude, periodic hills (Re = 5,600).

For the periodic hill flow case, DNS and highly-resolved LES data from Breuer et al. [15] are used. The flow is simulated using DNS at Reynolds numbers ranging from Re = 700 to 5,600 and using LES at Re = 10,595, based on the bulk velocity at the inlet and the hill height. The flow case was first proposed by Mellen et al. [86], as a simple geometry featuring both separation from a curved surface and reattachment on a flat plate. It was investigated by Fröhlich et al. [39], after which it was further studied for a broader range of Reynolds numbers in the work of which the data is used here. The flow case has been used to study the physical mechanisms of separation and to improve the performance of various turbulence models [2, 57, 155]. The study of flow separation over curved surfaces is relevant, as it occurs in many flows of practical interest [134]. Furthermore, the geometry has well-defined boundary conditions and computations can be performed relatively affordably.

On the upper and lower walls, no-slip boundary conditions are imposed. Similarly to the square duct flow case, a source term is added to the momentum equation and cyclic boundary conditions are imposed on the inlet and the outlet. The number of cells in x and y directions for the case where Re = 5,600 are 140 and 130, respectively. For the Reynolds number of 10,595, the number of cells in the y-direction was selected to be 160, with the same ratio between cells in the two direction as for the lower Reynolds number case. Again, a mesh convergence study was performed by varying the number of cells while keeping the aforementioned ratio the same. On both the upper and lower walls, the value of y^+ was kept below 1. The results of the mesh convergence study for the Re = 10,595 are shown in fig. 5.5.

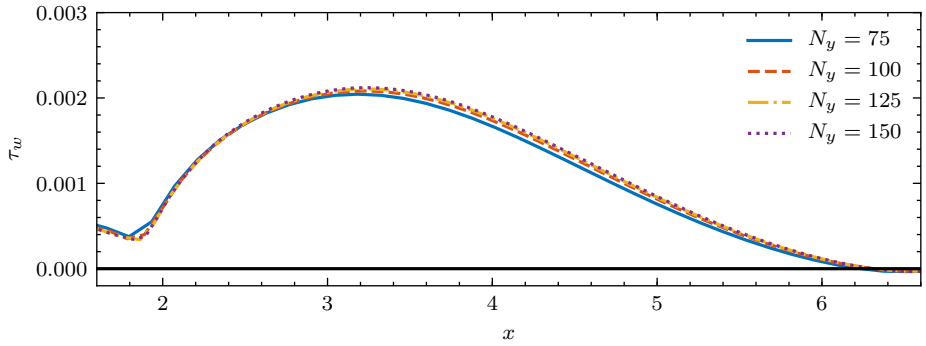


Figure 5.5: Wall shear stress for various mesh refinements, periodic hills (Re = 10,595).

The deficiencies in the predictions of RANS models in flows with a large separation region have been studied extensively [2, 58, 90, 129]. The $k-\omega$ model usually underpredicts the eddy viscosity in the shear layer, which causes insufficient mixing. Because of this, the flow reattaches too late and the separation region is excessively large. Multiple ad-hoc fixes to this behavior have been proposed over the years. One example is the work of Rumsey [129], in which a multiplier to the destruction term in the ω-equation is introduced as a function of the production to dissipation ratio. The desired behavior of this term is to increase the eddy viscosity in the shear layer such that the flow reattaches earlier.

5.4. Converging-diverging channel

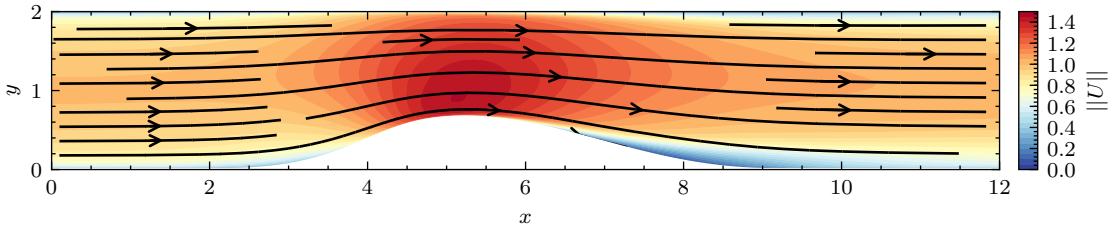


Figure 5.6: Mean velocity magnitude, converging-diverging channel (Re = 12,600).

Data for the converging-diverging channel case has been obtained from the study performed by Laval and Marquillie [76]. The flow is calculated using DNS at a Reynolds number of 12,600, based on the channel half-height and the maximum velocity at the inlet. The flow features a small separation bubble on the leeward side of the hill. No-slip boundary conditions are imposed on the upper and lower walls. In order to obtain the velocity and turbulent quantities at the inlet, a boundary layer was simulated using the same base model (the $k-\omega$ model) and the same Reynolds number. At the outlet, a zero gradient boundary condition is imposed on all variables except the pressure, which is fixed at a reference pressure. The mesh used in the field inversion and machine learning phases has 125 cells in the y-direction and 600 cells in the streamwise direction. A mesh convergence study was performed by varying the number of cells and keeping the ratio between the number of cells in both directions the same. The results of the mesh convergence study are shown in fig. 5.7. The separation and reattachment locations found in the DNS study are 5.8 and 6.6, respectively [76], and are indicated in the figure. For a variety of RANS turbulence models, Campos et al. [17] reports separation locations varying from 5.8 to 6.4 and reattachment locations varying from 9.1 and 9.6. In this case, the RANS model is also highly inaccurate in its prediction for the separation location and the bubble length.

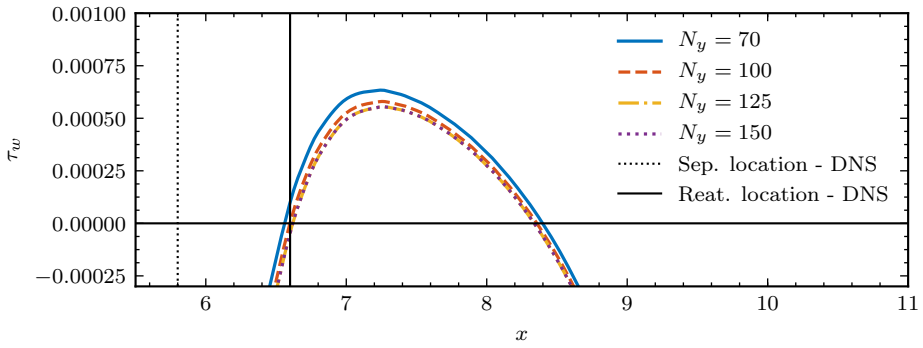


Figure 5.7: Wall shear stress for various mesh refinements, converging-diverging channel (Re = 12,600).

5.5. Backward facing step

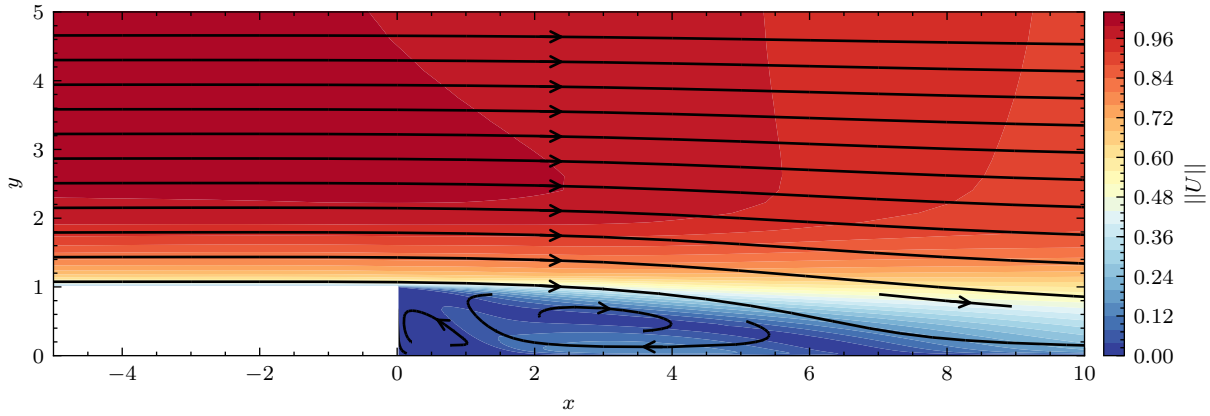


Figure 5.8: Mean velocity magnitude, backward facing step (Re = 5,100).

High-fidelity data for the backward-facing step was obtained from the study performed by Le et al. [77]. The flow was simulated using DNS at Re = 5,100 based on the step height and the inlet velocity. Compared to the periodic hills and backward facing step flow cases, the separation location is fixed in this case. Furthermore, the case features a more stronger separation than the aforementioned cases. In addition to the larger clockwise-rotating separated region, a small region in the corner rotates counter-clockwise.

On the lower boundary, a no-slip boundary condition is applied. Similarly to the converging-diverging channel, a boundary layer at the same Reynolds number was used for the inflow conditions. At the outlet, the same boundary conditions are applied as for the converging-diverging channel. On the upper wall, a slip boundary condition is imposed. The mesh was created with 90 cells in the y-direction and 225 cells in the streamwise direction. A mesh convergence study was performed by varying the number of cells in the y-direction and keeping the ratio between the number of cells in both directions the same. For all meshes in the convergence study, the value of y^+ was kept below 1 everywhere on the lower wall. The wall shear stress is shown in fig. 5.9, for various mesh refinements. Between the two finest grids, the difference in reattachment location is small. The reattachment location found in the DNS study by Le et al. [77] is shown to be at $x = 6.0$. Again, it can be noted that the $k-\omega$ model overpredicts the size of the separated region considerably.

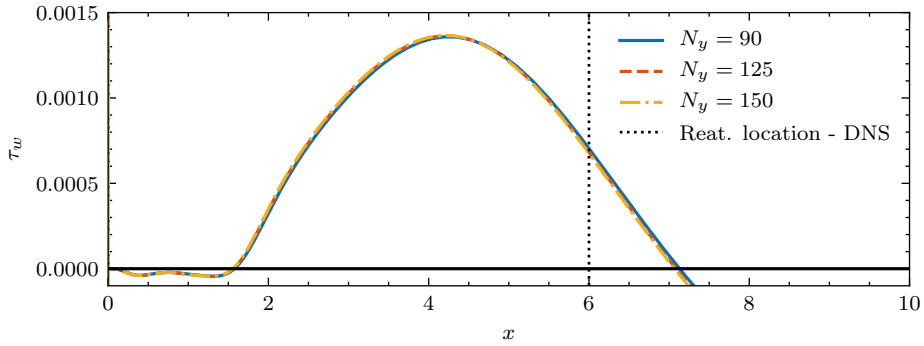


Figure 5.9: Wall shear stress for various mesh refinements, backward facing step (Re = 5,100).

6 Methodology

This chapter describes the details of the implementation of the paradigm of field inversion and machine learning. The steps involved in solving the forward model, i.e. the $k-\omega$ turbulence model, are given in section 6.1. In section 6.2, the two formulations of the corrective term are described. The adjoint equations and adjoint boundary conditions and their implementation in OpenFOAM are given in section 6.3. The details of the two phases of the paradigm are given in section 6.4 for the field inversion and section 6.5 for the machine learning phase. Finally, the computational cost of the full paradigm is briefly touched upon in section 6.6.

6.1. Base model

6.1.1. Governing equations and solution method

In this work, the $k-\omega$ model was selected as the baseline model. Because of the general nature of the paradigm of FIML, it can be applied to a broad range of physical models (with certain restrictions on the differentiability of the governing equations). Therefore, in the context of turbulence modeling, the choice for the $k-\omega$ model was mainly based on practical considerations. First of all, a known issue of the adjoint equations is the numerical stability [130]. As the adjoint equations have a form similar to the primal equations, it was expected to be convenient to choose a numerically stable baseline model. Therefore, an eddy viscosity model has been chosen over a Reynolds stress model. Secondly, the $k-\omega$ model has simple governing equations and can be directly applied in a low Reynolds number setting. Therefore, the difficulties of deriving an adjoint wall model such as in Zymaris et al. [173] can be avoided.

The standard high-Reynolds number $k-\omega$ turbulence model as described by Wilcox et al. [163] is used as the base model, with the standard coefficients as shown in table 6.1. The RANS equations with the $k-\omega$ model equations are given by

$$R_i^v = \frac{\partial(v_i v_j)}{\partial x_j} + \frac{\partial p}{\partial x_i} + \frac{2}{3}\frac{\partial k}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu+\nu_t)\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\right) = 0, \tag{6.1}$$

$$R^p = -\frac{\partial v_j}{\partial x_j} = 0, \tag{6.2}$$

$$R^k = \frac{\partial(v_j k)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial k}{\partial x_j}\right) - P_k + \frac{2}{3}\frac{\partial v_j}{\partial x_j}k + C_\mu \omega k = 0, \tag{6.3}$$

$$R^\omega = \frac{\partial(v_j \omega)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega \nu_t + \nu)\frac{\partial \omega}{\partial x_j}\right) - \gamma P + \frac{2}{3}\gamma\frac{\partial v_j}{\partial x_j}\omega + \alpha\omega^2 = 0, \tag{6.4}$$

where

$$P_k = \nu_t P = \nu_t \frac{\partial v_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right). \tag{6.5}$$

The open-source `C++` solver OpenFOAM [63] is used for all flow calculations. The SIMPLE algorithm [108] is used for pressure-velocity coupling. This algorithm is implemented in the `simpleFoam` solver in OpenFOAM. The `kOmega` turbulence model is used for the baseline simulations and is used as a basis for the adjoint turbulence model. All convective terms were discretized using second-order schemes. Some studies suggest

the use of first-order upwind schemes for the convective term in the adjoint momentum equation [50] in order to improve the convergence of the adjoint equations. However, this did not have a considerable effect for the flow cases considered in this work. In order to enhance the diagonal dominance of the steady state calculations, under-relaxation is used for all equations.

The term in the momentum equation including the gradient of the turbulent kinetic energy originated from the fact that, in OpenFOAM, the isotropic part of the Reynolds stress tensor is included in the pressure, resulting in a modified pressure given by

$$p^* = p + \frac{2}{3}k. \tag{6.6}$$

In the expressions presented here, p corresponds to the mean pressure, i.e. the isotropic term is written explicitly.

C_μ	α	γ	σ_k	σ_ω
0.09	0.072	0.52	0.5	0.5

Table 6.1: Coefficients used in the base and augmented $k-\omega$ model.

6.1.2. Boundary conditions

At solid walls, a no-slip boundary condition is imposed on the velocity and a zero Neumann boundary condition is imposed on the pressure. The turbulent kinetic energy is set to a low value ($k_{\text{wall}} = 10^{-15}$) in order to avoid division by zero. The boundary condition for ω becomes singular at the wall. As suggested by Menter [87], the value at the wall is set as

$$\omega_{\text{wall}} = \frac{60\nu}{\beta d^2}, \tag{6.7}$$

where d is the spacing of the first grid cell and $\beta = 0.075$. For all flow cases, the spacing of the first grid cell was chosen such that $y^+ < 1$ was satisfied along all solid walls. The boundary conditions at the outlet for the non-periodic cases were set to zero Neumann, except for the pressure, which was set to zero. Because only the pressure gradient occurs in the governing equations, its absolute value can be set to an arbitrary value at an arbitrary reference location.

6.2. Corrective term formulations

In this work, two formulations for the corrective term are considered. Firstly, the form used in [107, 136, 138] in the context of turbulence modeling is analyzed, i.e. one single scalar corrective field multiplying the production term of the ω equation. Secondly, two scalar corrective functions are defined as corrections to the eigenvalues of the anisotropy tensor. In addition, a preliminary investigation into the derivation of inferring a full description of the Reynolds stress tensor (i.e. inferring corrections to the eigenvectors in addition to the magnitude and the shape) will be given in appendix B.6.

In the first formulation, the corrective term is chosen to be a multiplication of the production term of the ω-equation:

$$R^{\omega\prime} = \frac{\partial(v_j\omega)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega \nu_t + \nu)\frac{\partial \omega}{\partial x_j}\right) - \beta_{\text{P}}\gamma P + \frac{2}{3}\gamma\frac{\partial v_j}{\partial x_j}\omega + \alpha\omega^2 = 0. \tag{6.8}$$

This choice changes the complete balance the turbulence model. As shown by Poroseva and Murman [117], it is the complete balance between all terms in a turbulence model which determines the performance of the model, not the accuracy of one specific term. Also, even if an optimal distribution of the corrective term was found, the predictions would still be limited by the linear eddy viscosity assumption used by the base model. These problems can be improved upon by applying the corrective term to an expression directly determining the Reynolds stress tensor. This would take away the need to apply the corrective term to a single term of a functional form of a turbulence model while at the same time allow for moving away from linear eddy viscosity models (LEVMs). Therefore, the second formulation of the corrective term considered in this work is applied to the eigendecomposition of the anisotropy tensor.

There are several options for the corrective function which allow to correct the full anisotropy tensor, i.e. its magnitude, shape, and orientation. One option would be to infer the ten scalar coefficient of the tensor basis used by Ling et al. [81]. This would result in a Galilean invariant representation of the Reynolds stress tensor. However, as the anisotropy tensor is characterized by six quantities (one for the magnitude, two for the eigenvalues, and three for the eigenvectors), this formulation would introduce additional redundancy

into the problem. A more experimental option would be to infer source terms in the transport equations for the barycentric coefficients proposed by Edeling et al. [34]. However, by introducing more primal equations, this would also result in more adjoint equations. Furthermore, this would complicate bounding the inferred quantities and might further deteriorate the numerical stability of the solver. Finally, the corrective terms could be formulated as direct corrections to the magnitude, shape, and orientation of the anisotropy tensor. Representing the eigenvector correction as a three-dimensional rigid body rotation using unit quaternions ensures the realizability of the corrected anisotropy tensor. Furthermore, this will result in smoother functions than other representations and be frame-independent [166]. For the eigenvalues, the barycentric coordinates can be used to obtain an interpretable correction which can be bounded.

The eigendecomposition of the anisotropy tensor is given by $b = V\Lambda V^T$, where V is a matrix of which the columns are the eigenvectors of b and $\Lambda \equiv \text{diag}(\lambda_1, \lambda_2, \lambda_3)$. Then, the corrected anisotropy tensor is given by

$$b' = V(\Lambda + B)V^T = V\Lambda V^T + VBV^T = b + VBV^T, \tag{6.9}$$

where $B \equiv \text{diag}(\Delta\lambda_1, \Delta\lambda_2, \Delta\lambda_3)$. In order for the corrective term to have a more interpretable meaning, the eigenvalues are corrected by considering corrections in their barycentric coefficients, $\beta_{C_{1c}}$ and $\beta_{C_{2c}}$ instead of inferring corrections to the eigenvalues directly. Note that only two functions have to be inferred due to the restriction that the barycentric coefficients sum to one. From the explanation in section 2.3.2, it can be derived that

$$B = \text{diag}\,(\Delta\lambda_1, \Delta\lambda_2, \Delta\lambda_3) = \text{diag}\left(\frac{2}{3}\beta_{C_{1c}} + \frac{1}{6}\beta_{C_{2c}}, -\frac{1}{3}\beta_{C_{1c}} + \frac{1}{6}\beta_{C_{2c}}, -\frac{1}{3}\beta_{C_{1c}} - \frac{1}{3}\beta_{C_{2c}}\right). \tag{6.10}$$

This correction affects both the momentum and the production term of the k-equation:

$$R_i^{v'} = \frac{\partial(v_i v_j)}{\partial x_j} + \frac{\partial p}{\partial x_i} + \frac{2}{3}\frac{\partial k}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) + 2\frac{\partial}{\partial x_j}\left(kV_{in}B_{nl}V_{jl}\right) = 0, \tag{6.11}$$

$$R^{k'} = \frac{\partial(v_j k)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial k}{\partial x_j}\right) - P_k + 2kV_{in}B_{nl}V_{jl}\frac{\partial v_i}{\partial x_j} + \frac{2}{3}\frac{\partial v_j}{\partial x_j}k + C_\mu \omega k = 0. \tag{6.12}$$

Again, it should be noted that using two spatially varying corrective terms instead of one has practically no effect on the computational cost when an adjoint method is used to obtain the gradients.

The possibilities of inferring the complete representation of the Reynolds stress tensor, i.e. the magnitude, shape, and orientation, has also been investigated. This considerably complicates the derivation of the continuous adjoint. This is because, in contrast to just inferring the eigenvalues, the correction cannot be added as a source term to the momentum and k-equation when the corrections to the eigenvectors are inferred in addition. The preliminary results of this investigation are described in appendix B.6.

6.3. Continuous adjoint of the $k-\omega$-model

6.3.1. Governing equations

The derivation of the continuous adjoint for the $k-\omega$ turbulence model follows the same procedure as was described for the general case in section 3.4.2, and follows, to some extent, the derivation of the continuous adjoint for the $k-\omega$ SST model presented by Kavvadias et al. [71]. The solution to the field inversion is found by minimizing the negative of (3.7). In this work, diagonal covariance matrices with constant variance are used. The objective function is then given by

$$J = \frac{1}{2\sigma_m^2}\sum_{i=1}^{N}(d_i - h(\beta)_i)^2 + \sum_{j=1}^{M}\frac{1}{2\sigma_\beta^2}(\beta_j - \beta_{\text{prior},j})^2. \tag{6.13}$$

A detailed derivation of the adjoint equations and boundary conditions can be found in appendix B. For the sake of brevity, only the resulting equations are given here:

$$R_i^u = \frac{\partial J_\Omega}{\partial v_i} - \frac{\partial u_i v_j}{\partial x_j} - v_j \frac{\partial u_j}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu+\nu_t)\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right) + \frac{\partial q}{\partial x_i} - k\frac{\partial k_a}{\partial x_i} - \omega\frac{\partial \omega_a}{\partial x_i} \tag{6.14}$$
$$+2\frac{\partial}{\partial x_j}\left((\omega_a\gamma + k_a\nu_t)\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\right) - \frac{2}{3}\gamma\frac{\partial(\omega_a\omega)}{\partial x_i} - \frac{2}{3}\frac{\partial(k_a k)}{\partial x_i} = 0,$$

$$R^q = \frac{\partial J_\Omega}{\partial p} - \frac{\partial u_i}{\partial x_i} = 0, \tag{6.15}$$

$$R^{k_a} = \frac{\partial J_\Omega}{\partial k} - \frac{\partial(k_a v_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_k\nu_t+\nu)\frac{\partial k_a}{\partial x_j}\right) + \frac{\partial k_a}{\partial x_j}\sigma_k\frac{\partial k}{\partial x_j}\frac{1}{\omega} + \frac{\partial \omega_a}{\partial x_j}\sigma_\omega\frac{\partial \omega}{\partial x_j}\frac{1}{\omega} + \frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{1}{\omega} \tag{6.16}$$
$$-k_a\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{1}{\omega} + \frac{2}{3}k_a\frac{\partial v_j}{\partial x_j} - \frac{2}{3}\frac{\partial u_i}{\partial x_i} + C_\mu k_a\omega = 0,$$

$$R^{\omega_a} = \frac{\partial J_\Omega}{\partial \omega} - \frac{\partial(\omega_a v_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega\nu_t+\nu)\frac{\partial \omega_a}{\partial x_j}\right) - \frac{\partial \omega_a}{\partial x_j}\sigma_\omega\frac{\partial \omega}{\partial x_j}\frac{k}{\omega^2} + \frac{2}{3}\gamma\omega_a\frac{\partial v_j}{\partial x_j} + 2\alpha\omega_a\omega - \frac{\partial k_a}{\partial x_j}\sigma_k\frac{\partial k}{\partial x_j}\frac{k}{\omega^2} \tag{6.17}$$
$$+k_a\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{k}{\omega^2} + C_\mu k_a k - \frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{k}{\omega^2} = 0.$$

It can be noted that the adjoint equations are similar to the primal equations, the main difference being the presence of several additional source terms. Furthermore, it can be seen that terms with a spatial derivative of odd order have switched signs whereas terms including an even-ordered spatial derivative have the same sign as in the primal equations. The similarity between the primal and the adjoint equations allows for a similar implementation, in this case in OpenFOAM.

When the misfit in the mean velocities is used in the likelihood, the only non-zero explicit derivative of the objective function with respect to a primal variable (the first term for each adjoint equation above) is

$$\frac{\partial J_\Omega}{\partial v_i} = \frac{1}{\sigma_m^2}(v_i - v_{\mathrm{DNS},i}). \tag{6.18}$$

Including the mean pressure in the objective function would complicate the implementation of the adjoint equations, as the divergence of the adjoint velocity would be nonzero.

The adjoint equations presented above do not yet include the correction to the base model. The changes which need to be made for the specific correction term formulations used in this work are also described in appendix B. The gradient with respect to the production term correction is then given by

$$\frac{\delta L}{\delta \beta} = -\int_\Omega \omega_a\gamma P\,\mathrm{d}\Omega + \frac{1}{\sigma_\beta^2}(\beta - \beta_{\mathrm{prior}}). \tag{6.19}$$

For the eigenvalue corrections, the adjoint gradient is given by

$$\frac{\delta L}{\delta \beta} = \int_\Omega 2u_i\frac{\partial}{\partial x_j}\left(kV_{in}\frac{\delta B_{nl}}{\delta \beta}V_{jl}\right)\mathrm{d}\Omega + \int_\Omega 2k_a k\frac{\partial v_i}{\partial x_j}V_{in}\frac{\delta B_{nl}}{\delta \beta}V_{jl}\,\mathrm{d}\Omega + \frac{1}{\sigma_\beta^2}(\beta - \beta_{\mathrm{prior}}). \tag{6.20}$$

These gradients can be calculated straightforwardly after the primal and adjoint equations are converged.

6.3.2. Boundary conditions

At the boundaries where Dirichlet conditions are imposed on the primal velocity, k, and ω, the boundary conditions for their adjoint counterparts are zero Dirichlet and zero Neumann for the adjoint pressure. At the outlet, however, the specifications for the adjoint boundary conditions are more complex. Again, the full derivation is given in appendix B. The boundary condition for the adjoint velocity is derived from the tangential component of the boundary integral including the sensitivity of the primal velocity:

$$\begin{aligned} &v_n u_{t,i} + n_j\frac{\partial u_{t,i}}{\partial x_j}(\nu+\nu_t)n_j - 2\left(\omega_a\gamma\beta + k_a\nu_t\right)n_j\frac{\partial v_{t,i}}{\partial x_j} \\ &\quad + \frac{\partial u_j}{\partial x_i}(\nu+\nu_t)n_j - \frac{\partial u_j}{\partial x_k}(\nu+\nu_t)n_j n_i n_k \\ &\quad - 2\left(\omega_a\gamma\beta + k_a\nu_t\right)\frac{\partial v_j}{\partial x_i}n_j + 2\left(\omega_a\gamma\beta + k_a\nu_t\right)\frac{\partial v_j}{\partial x_k}n_j n_i n_k = 0. \end{aligned} \tag{6.21}$$

The boundary condition for the adjoint pressure is derived from the normal component of the boundary integral including the sensitivity of the primal velocity, and is given by

$$\begin{aligned} q = {} & u_n v_n + u_j v_j + 2(\nu + \nu_t) n_j \frac{\partial u_n}{\partial x_j} + \omega_a \omega - 2\omega_a \beta\gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) n_i n_j + \frac{2}{3}\gamma\omega_a\omega + k_a k \\ & - 2k_a n_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + \frac{2}{3} k_a k. \end{aligned} \tag{6.22}$$

Finally, the boundary conditions for k_a and ω_a are derived from their corresponding boundary integrals:

$$k_a = \frac{-u_i n_j \left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i} \right) \frac{k}{\omega^2} - \omega_a n_j v_j - \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{k}{\omega^2} - \frac{\partial \omega_a}{\partial x_j} (\sigma_\omega \nu_t + \nu) n_j}{n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{k}{\omega^2}}, \tag{6.23}$$

$$\omega_a = \frac{-u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \frac{1}{\omega} + k_a v_j n_j - k_a n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{1}{\omega} + \frac{\partial k_a}{\partial x_j} (\sigma_k \nu_t + \nu) n_j}{n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{1}{\omega}}. \tag{6.24}$$

6.3.3. Hessian approximation

Similarly to the one-dimensional model problem in section 3.5, the Gauss-Newton approximation is used to approximate the posterior covariance. Differentiating the objective function in eq. (6.13) twice with respect to the corrective term gives

$$\frac{\delta^2 J}{\delta\beta_j \delta\beta_k} \approx \frac{1}{\sigma_m^2} \sum_{i=1}^{N} \frac{\delta v_i}{\delta\beta_j} \frac{\delta v_i}{\delta\beta_k} + \frac{1}{\sigma_\beta^2} \delta_{jk}, \tag{6.25}$$

given that the mean velocity is used as the data in the likelihood. The derivatives in the first term can be calculated using the adjoint equations with the following objective function:

$$J_i = \frac{v_i - v_{\mathrm{HF},i}}{\sigma_m}. \tag{6.26}$$

Changing the objective function requires changing the source term in the adjoint momentum equation and the expression of the partial derivative of the objective function with respect to the corrective term, i.e.

$$\frac{\partial J_i}{\partial \beta_j} = 0 \quad \text{and} \quad \frac{\partial J_i}{\partial v_j} = \frac{1}{\sigma_m} \delta_{ij}. \tag{6.27}$$

This approach requires solving the adjoint equations for the N objective functions J_i. In practice, this can be done relatively quickly using the converged adjoint variables as initial conditions.

6.3.4. OpenFOAM implementation

In order to implement the continuous adjoint equations, boundary conditions, and one-shot optimization method, a new OpenFOAM solver based on `simpleFoam` was written. The current adjoint solver in OpenFOAM is based on the work of Othmer et al. [102]. In this solver, a porosity term is introduced in the momentum equation. The main idea is that areas where the optimization yields low values of the porosity are more fluid-like than areas where the porosity is high. This implementation uses the one-shot approach for the optimization process. Furthermore, the frozen turbulence assumption is used. Therefore, the turbulence is assumed not to change explicitly with changes in the porosity value and there is no corresponding adjoint turbulence model.

The adjoint solver and adjoint turbulence model used in this work will be referred to as `adjointSimpleFoam` and `adjointkOmega`, respectively. Due to the similarity between the primal and the adjoint equations, a similar solution procedure could be used for the adjoint equations. The procedure is illustrated in fig. 6.1. Each `adjointSimpleFoam` iteration starts with solving the primal velocity and pressure, after which the adjoint velocity and pressure are solved. The equations for k_a and ω_a were implemented in existing `kOmega` model. Therefore, the algorithm proceeds with solving the primal equations for k and ω, after which their adjoint equivalents are solved. Now all adjoint fields are obtained, the adjoint gradients can be calculated. The corrective term is then updated using a simple gradient descent update rule, after which the objective function is calculated. The process is repeated until convergence of the objective function. Similarly to the primal equations, all adjoint equations use under-relaxation to enhance the stability

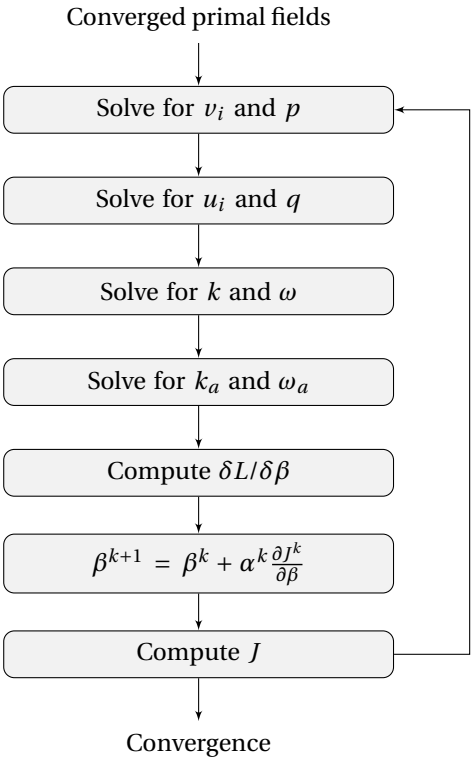


Figure 6.1: Flowchart of `adjointSimpleFoam`.

of the iterative solver. In most cases, the under-relaxation factors for the adjoint equations could be set to a value slightly lower than that of the primal equations. The most relevant sections of the code are given in appendix C.

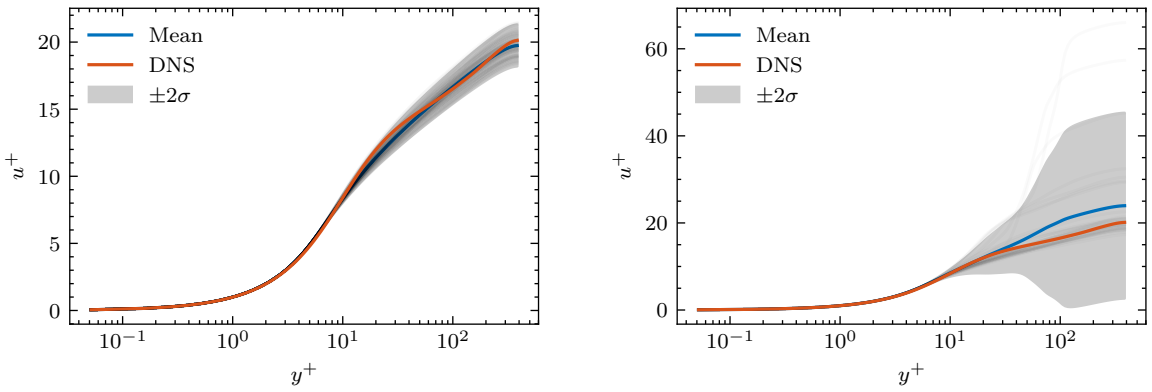
6.4. Field inversion

6.4.1. Implementation

For the complete approach, OpenFOAM was coupled with Python, in which the optimization routines implemented in Scipy [65] were used. In the tests performed with the complete approach in the optimization phase, the conjugate gradient method was used. Quasi-Newton methods such as L-BFGS did not provide an advantage in convergence or robustness compared to the conjugate gradient method for the cases considered. In the complete approach, the primal and adjoint equations were solved using the corrective function given by the optimization algorithm until a specified convergence criterion was reached. Then, the objective function and adjoint gradient were returned to the Scipy optimization algorithm. Therefore, the solution process as displayed in fig. 6.1 was used, with the one-shot gradient descent step size $\alpha^k = 0$. The convergence speed, accuracy, and stability of the optimization process employing the complete approach can be traded-off using the convergence criteria for the primal and adjoint equations. Setting the required minimum initial residual to a lower value results in more `simpleFoam` iterations for each step in the optimization. Therefore, the overall optimization procedure will take longer, but the gradients will be more accurate.

6.4.2. Parameter selection

For the field inversion phase, a decision needs to be made on whether the complete or one-shot approach is more suitable. Furthermore, the step size used in the optimization, the observational covariance, and the prior distribution need to be chosen adequately. First of all, the covariance in the observational data was neglected, as discussed before. The observational covariance is thus given by $C_m = \sigma_m^2 I$. Similarly to Parish and Duraisamy [107], the standard deviation was chosen to be $\sigma_m = 10^{-10}$, reflecting a high confidence in the accuracy of the high-fidelity data. The prior distribution is also chosen to be a multivariate Gaussian with a constant diagonal covariance matrix. The mean corresponds to the base model, i.e. for the production term



(a) Propagated prior samples, $\sigma_\beta = 0.5$.

(b) Propagated prior samples, $\sigma_\beta = 1$.

Figure 6.2: Example of prior selection, turbulent channel flow ($Re_\tau = 395$).

correction $\beta_{\text{prior}} = 1$ and for the eigenvalue correction $\beta_{\text{prior}} = 0$. The standard deviation was determined using the procedure proposed by Parish and Duraisamy [106]. Samples were taken from the prior using an initial estimate of the variance. Then, these samples were propagated through the forward model. It is then checked whether the quantity of interest of the high-fidelity data falls within the $\pm 2\sigma$ bounds resulting from the prior samples. This process is repeated until this is satisfied. The method is illustrated in fig. 6.2. Where $\sigma_\beta = 1$, the streamwise velocity profile from the DNS data falls within the uncertainty bounds with an excessively large margin. When $\sigma_\beta = 0.5$, this margin is considerably smaller, and the discrepancy in the buffer layer drives the decision for the prior covariance.

The gradient descent step size was found for each case and correction term formulation by varying it over a range of values of which the highest results in divergence and the lowest in (slow) convergence, as it is expected that the optimal step size lies between these two extremes. Higher step sizes can lead to numerical instabilities in the optimization process, as the accuracy of the adjoint gradient worsens. Below the value of the step size for which the optimization was stable, the chosen value did not have an effect on the result of the optimization process. The numerical stability of the optimization process was also improved using converged primal and adjoint fields as initial condition. The effect of the gradient descent step size and convergence criteria on the convergence of the optimization process is further discussed in section 7.1.2. In order to decide whether to use the complete approach or the one-shot approach in the optimization phase, both approaches were tested for a range of values of the one-shot gradient descent step size and the required minimum initial residuals. Both parameters have a similar effect on the overall optimization process, i.e. trading off accuracy, convergence, and stability. The results of this study will be discussed and further interpreted in section 7.1.2 for the converging-diverging channel flow case.

6.5. Machine learning

6.5.1. Implementation

The machine learning phase aims to approximate a function between the RANS features and the MAP results for a number of flow cases. As discussed in section 2.5, two approaches can be distinguished in the way the machine learning results are used in the RANS solver. In the first approach, the corrective approach, the machine learning algorithm output is propagated through the flow solver once. In the iterative approach, the algorithm is called in every iteration of the flow solver. So far, the corrective approach is still the most common approach in literature [81, 168]. Investigating whether an iterative approach can converge and whether the converged solution is accurate is an important topic for future research.

A short study was performed on the feasibility of predicting the corrective term every iteration of the flow solver. A feedforward neural network with two hidden layers with 64 nodes each was trained on the square duct flow case at four different Reynolds numbers ranging from Re = 1,100 to Re = 2,900. The neural network was then used to predict the corrective term in an iterative manner for a higher Reynolds number (Re = 3,500). This problem setting already proved to yield a considerable improvement when the corrective approach was used with the same neural network. However, in the iterative setting this approach diverged after a low number of iterations. It was attempted to improve this behavior using relaxation for the update of

the corrective term, i.e. updating it according to

$$\beta^k = (1-\alpha)\hat{\beta}^{k-1} + \alpha\hat{\beta}^k, \tag{6.28}$$

at iteration k, where $\hat{\beta}$ is the output of the neural network and $\alpha \in [0,1]$ is the relaxation factor. However, this did not yield significant improvements in the numerical stability. Therefore, in this work, the corrective approach is used. In cases where propagating the corrective term results in problematic numerical behavior, it is also possible to use blending. For example, Duraisamy et al. [29] used the blending function

$$\beta = (1-\alpha)\beta_{\text{baseline}} + \alpha\beta_{\text{ML}}, \tag{6.29}$$

to alleviate numerical problems. The blending factor α can then be gradually increased from 0 to 1, in which β_{ML} is kept constant. The propagations used in this work did not cause any numerical problems and therefore no blending was used in the machine learning phase. Parish and Duraisamy [107] describe using an additional Bayesian update step in order to improve the results in regions where the machine learning predictions are inaccurate. The final step is then given by

$$\beta_{\text{post}} = \arg\min_{\beta}\left(-\frac{1}{2}(\beta-\beta_{\text{ML}})^T C_{\beta_{\text{ML}}}^{-1}(\beta-\beta_{\text{ML}})) - \frac{1}{2}(\beta-\beta_{\text{prior}})^T C_{\beta}^{-1}(\beta-\beta_{\text{prior}})\right). \tag{6.30}$$

In this optimization process moves the corrective term closer to the machine learning solution in areas where the uncertainty is low. In other regions, the corrective term will be closer to the base model.

The training and prediction of the machine learning algorithm is done in Python, using the Scikit-learn package [109] for the random forests and Gaussian processes. For preliminary experiments with neural networks, TensorFlow [1] was used. In the corrective approach, the output of the machine learning algorithms was written to the OpenFOAM case file, after which it was propagated using the `adjointSimpleFoam` solver with zero step size. For the iterative approach, an open-source library called `frugally-deep` [51] was used. This library allows using a neural network trained in Python (using Keras [22]) to obtain predictions in `C++`. The advantage of this method is that it allows flexible experimentation in Python for the design of the neural network architecture and hyperparameter tuning, while only the prediction is done in `C++`. This method was implemented in the adjoint turbulence model `adjointkOmega`, where the features were calculated every iteration and the corrective term was obtained from the neural network.

6.5.2. Algorithm choice

Preliminary investigations were performed in order to select the machine learning algorithm to be used in the second phase of the paradigm. On the basis of the results from literature, three classes of algorithms were tested: Gaussian processes, feedforward neural networks, and random forests. It was found that the specific choice of machine learning algorithm did not have a significant effect on the accuracy of the results. This finding is also supported by literature, where the main improvements in accuracy in a purely machine learning based approach were made by incorporating physical principles as invariance in the algorithms [68, 81]. The use of these kinds of algorithms is beyond the scope of this work. Therefore, the choice of machine learning algorithm is mainly driven by reasons related to the implementation, speed, and interpretability of the results. For this reason, Gaussian processes were used for the turbulent channel flow experiments. For the other flow cases, Gaussian processes could not be used due to the higher number of grid cells, resulting in a larger dataset and thus higher memory requirements. For these cases random forests were used.

6.5.3. Machine learning features

This work uses the same features as used in the thesis of Kaandorp [67]. An overview of the features is given in table 6.2. The first two feature sets make use of the features proposed by Wang et al. [158]. Feature set 1 contains the features based on the normalized shear rate tensor and rotation rate tensor, which are given by

$$\hat{S}_{ij} = \frac{1}{2}\frac{k}{\varepsilon}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \quad \text{and} \quad \hat{\Omega}_{ij} = \frac{1}{2}\frac{k}{\varepsilon}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right). \tag{6.31}$$

Feature set 2 contains the features in which, in addition to the normalized shear rate tensor and strain rate tensor, the gradient of the turbulent kinetic energy is used. The gradient of the turbulent kinetic energy is first normalized using $\sqrt{k}/\varepsilon$. The tensor $\boldsymbol{A}_k$ is constructed by mapping the gradient of the normalized turbulent kinetic energy to an anti-symmetric tensor, i.e.

$$\boldsymbol{A}_k = \boldsymbol{I} \times \nabla\overline{k}, \tag{6.32}$$

Feature set	Features	Normalization	Notes
FS1	$\hat{\boldsymbol{S}}^2$, $\hat{\boldsymbol{S}}^3$, $\hat{\boldsymbol{\Omega}}^2$, $\hat{\boldsymbol{\Omega}}^2\hat{\boldsymbol{S}}$, $\hat{\boldsymbol{\Omega}}^2\hat{\boldsymbol{S}}^2$, $\hat{\boldsymbol{\Omega}}^2\hat{\boldsymbol{S}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{S}}^2$	-	
FS2	$\boldsymbol{A}_{\boldsymbol{k}}^2$, $\boldsymbol{A}_{\boldsymbol{k}}^2\hat{\boldsymbol{S}}$, $\boldsymbol{A}_{\boldsymbol{k}}^2\hat{\boldsymbol{S}}^2$, $\boldsymbol{A}_{\boldsymbol{k}}^2\hat{\boldsymbol{S}}\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}^2$, $\hat{\boldsymbol{\Omega}}\boldsymbol{A}_{\boldsymbol{k}}$, $\hat{\boldsymbol{\Omega}}\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}$, $\hat{\boldsymbol{\Omega}}\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}^2$, $\hat{\boldsymbol{\Omega}}^2\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}^*$, $\hat{\boldsymbol{\Omega}}^2\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}^{2*}$, $\hat{\boldsymbol{\Omega}}^2\hat{\boldsymbol{S}}\boldsymbol{A}_{\boldsymbol{k}}\hat{\boldsymbol{S}}^{2*}$	-	
FS3	$\frac{1}{2}\left(\|\|\hat{\Omega}_{ij}\|\|^2 - \|\|\hat{S}_{ij}\|\|^2\right)$ k $\min\left(\frac{\sqrt{k}d}{50\nu}, 2\right)$ $v_k\frac{\partial p}{\partial x_k}$ $\frac{k}{\varepsilon}$ $\sqrt{\frac{\partial p}{\partial x_i}\frac{\partial p}{\partial x_i}}$ $v_i\frac{\partial k}{\partial x_i}$ $\|\|R_{ij}\|\|$ $\left\|v_i v_j \frac{\partial v_i}{\partial x_j}\right\|$	$\|\|\hat{S}_{ij}\|\|^2$ $\frac{1}{2}v_i v_i$ - $\sqrt{\frac{\partial p}{\partial x_j}\frac{\partial p}{\partial x_j}v_i v_i}$ $\frac{1}{\|\|S\|\|}$ $\frac{1}{2}\frac{\partial v_k^2}{\partial x_k}$ $\left\|R_{jk}S_{jk}\right\|$ k $\sqrt{v_l v_l v_i \frac{\partial v_i}{\partial x_j} v_k \frac{\partial v_k}{\partial x_j}}$	

Table 6.2: Features used in the machine learning phase (Kaandorp [67]). FS1 and FS2 originate from [158], FS3 from Wu et al. [168]. For FS1 and FS2, the trace of the indicated tensors is taken. The asterisk (*) denotes that the trace of the indicated term should be taken for all cyclic permutations of the anti-symmetric tensor.

where $\boldsymbol{I}$ is the rank 2 identity tensor and $\overline{k}$ is the normalized turbulent kinetic energy. The third feature set contains a number of ad-hoc features with a physical interpretation and are obtained from Wu et al. [168]. For example, the first feature in this set is the Q-criterion. These features are normalized according to the normalization factors indicated in table 6.2. It should be noted that not all features in this set are Galilean invariant, as they include the pressure gradient and the velocity. For the cases considered in this work, this was not problematic. However, when the difference between the training and test flow cases is larger these features should be removed or reformulated.

When Gaussian processes are used for the machine learning case, the features are scaled to have zero mean and unit variance. For the cases where random forests are used, scaling is not necessary. For each case, multiple features are zero or contain little information because of a noisy pattern. Therefore, features in the training set with a variance lower than 10^{-4} were not used in the training.

6.5.4. Hyperparameter tuning

Random forest Before using the random forest in the machine learning phase, the effect of three hyperparameters on the root-mean-square error of the predicted corrective term will be studied. The study is performed using k-fold cross-validation with $k = 5$, on a database consisting of the inferred fields of the periodic hills flow case at Re = 5,600 and Re = 10,595 and the converging-diverging channel flow case at Re = 12,600. It should be noted that each hyperparameter was varied while keeping the others constant. This does not take into account the effect of the interaction of the hyperparameters on the test error of the algorithm. However, for the hyperparameters considered in this section, a similar general trend can be expected for other combinations.

The first hyperparameter is the number of estimators to use in the ensemble model. More trees generally result in a better performance, but increase the time required to train the model. Furthermore, the strongest performance improvement is usually observed below a certain critical number of trees, after which adding more trees does not yield better results. The results of the study are shown in fig. 6.3. Indeed, the error decreases most up until $N_{\text{estimators}} = 20$, after which the decrease in error is fairly low. As the computational time for the machine learning experiments in this work using random forests was relatively low, a number of estimators of $N_{\text{estimators}} = 80$ was chosen. The second hyperparameter to be investigated is the number of features in the subset of the total feature set to consider during splitting. A low number of features leads to more diverse trees, yielding a reduction in variance but an increase in bias. Therefore, it is important to cross-validate the effect of this hyperparameter. Literature suggests setting the hyperparameter equal to the number of features divided by three with a minimum of five for regression problems [38] and equal to the square root of the number of features with a minimum of one for classification problems. The effect of this

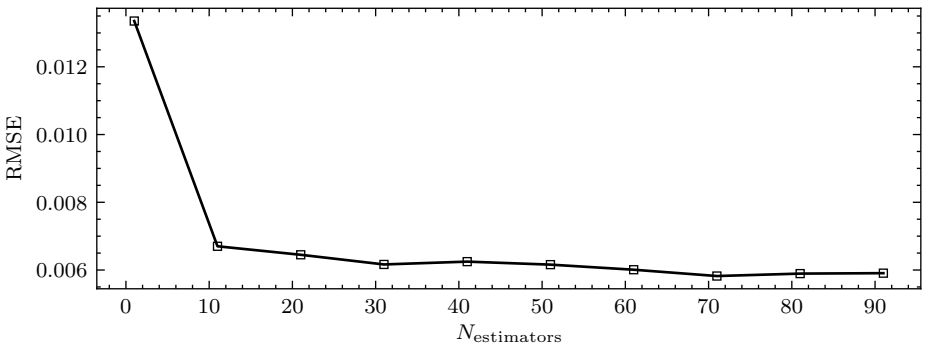


Figure 6.3: Mean effect of the number of estimators on predictive accuracy.

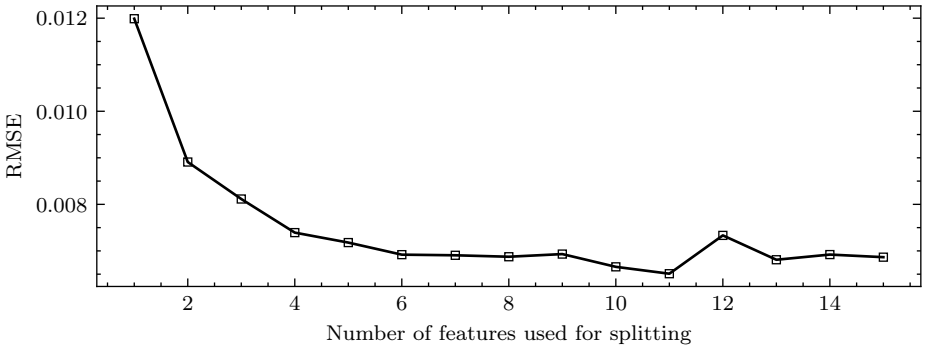


Figure 6.4: Mean effect of the number of features selected in each split on predictive accuracy.

hyperparameter is shown in fig. 6.4. The total number of features in this experiment was 15. The estimation of the test error is lowest for $N_{\text{features,split}} = 11$. Therefore, this value was chosen for the machine learning phase. Finally, the third hyperparameter considered was the minimum number of samples required in each of the branches in order for the split to be performed. A higher number of samples has a smoothing effect, which might be desirable in this specific application.

Gaussian process There are various approaches to model selection when using Gaussian processes. Choices to make in the model selection are usually confined to the selection of the kernel and its hyperparameters [125]. In contrast to other machine learning algorithms, e.g. neural networks, the number of hyperparameters to tune in a Gaussian process is relatively low. Similarly to the hyperparameter selection for the random forest, it is possible to do a grid search or to vary one hyperparameter at a time. However, in this work an empirical Bayes approach is taken, which enables the use of optimization techniques. As introduced in section 4.2, it is common to maximize the log marginal likelihood. The derivative with respect to the kernel

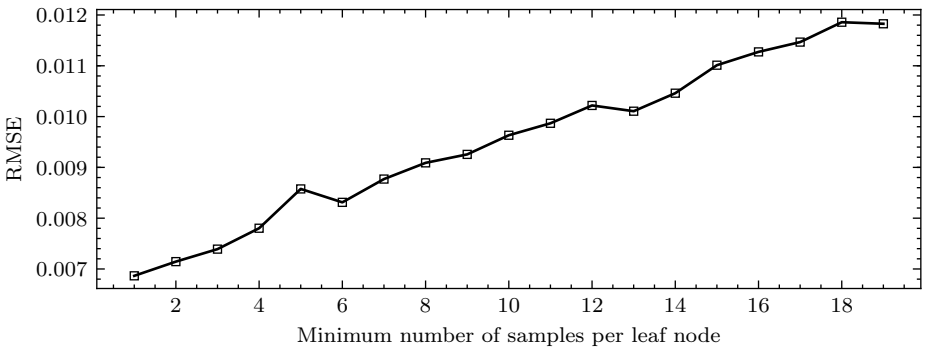


Figure 6.5: Mean effect of the required minimum number of samples for splitting.

parameters can be derived to be

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y}|\boldsymbol{X}) = \frac{1}{2} \boldsymbol{y}^T \boldsymbol{K}_y^{-1} \frac{\partial \boldsymbol{K}_y}{\partial \theta_j} \boldsymbol{K}_y^{-1} \boldsymbol{y} - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{K}_y^{-1} \frac{\partial \boldsymbol{K}_y}{\partial \theta_j} \right). \tag{6.33}$$

Then, a gradient-based optimization algorithm can be used to find the optimal kernel parameters. In this case, the L-BFGS-B algorithm is used. As multiple local optima might be present, the optimization is started several times, starting from randomly sampled kernel parameters.

An alternative approach to selecting the hyperparameters in a Gaussian processes is using Bayesian inference. Another alternative is multiple kernel learning, in which the weights of a weighted sum of base kernels are optimized instead of the kernel parameters [122].

6.6. Computational cost

Given the various phases and subphases of the paradigm, it might be convenient to shortly discuss the computational cost of each step in the method. The largest part of the computational cost of the paradigm is associated with the field inversion phase. First of all, sampling the prior to determine the prior covariance requires a relatively high number of forward solves. Sampling is also necessary to determine the posterior distribution of the quantities of interest and, if a stochastic machine learning algorithm is used, to obtain an estimate of the uncertainty of the machine learning output. The higher the desired accuracy of the mean and the variance, the more samples are required. However, the sampling lends itself conveniently to parallelization [107]. The calculation of the approximate Hessian requires N adjoint solves. This cost is excessively high, even for the simplest two-dimensional cases.

The computational cost of the optimization performed in the field inversion phase depends, among other factors, on the numerical stability of the considered case. In the most convergent cases, the optimization using the one-shot approach could be performed in a number of iterations comparable to one flow calculation, given that the optimization started with converged primal and adjoint fields. In other cases, the convergence of the optimization could take a considerably higher number of iterations, sometimes of $\mathcal{O}(100)$ forward model computations. It should be noted that the efficiency of the optimization phase also depends on adequate tuning of the relaxation parameters of the primal and adjoint equations, the step size in the one-shot optimization, or the convergence criterion in the complete approach. Usually, several runs are required to find the optimal settings.

A third phase to consider is the training of the machine learning algorithm. The computational cost associated with this process heavily depends on the choice of machine learning algorithm, the size of the flow cases included in the training set, and the number of features used. For example, when neural networks were used, the training took a longer time than when random forests or Gaussian processes were used. It should be noted that all sources of computation time discussed so far are all performed offline. The only online process in the paradigm is the prediction using the machine learning algorithm. For most algorithms, the cost of this step is low.

7

Results - field inversion

This chapter presents the results of the field inversion phase. In section 7.1, the gradients resulting from the adjoint solver are verified. Furthermore, the stability, robustness, and convergence of the one-shot and complete approach are compared. Section 7.2 describes the inferred corrective terms and the resulting mean velocities and other quantities of interest. The main questions to be answered in this chapter are whether the corrective terms allow the quantities of interest to be inferred correctly and whether the results have a physical interpretation.

7.1. Verification of the adjoint solver

7.1.1. Gradient verification

Similarly to the model problem considered in chapter 3 and chapter 4, the gradients resulting from the adjoint solver are verified using finite differences before using the solver in the optimization phase. The gradients resulting from the two methods are shown in fig. 7.1 for the turbulent channel flow case at a Reynolds number of $\mathrm{Re}_\tau = 590$. The gradients are evaluated at a production term correction corresponding to the base model, i.e. $\beta_\mathrm{P} = 1$. The gradient calculated using the adjoint variables is practically equal to that resulting from finite differences at all values of y. Furthermore, the finite difference gradients for both step sizes also coincide. This suggests that the step size is selected appropriately.

In order to verify that the impact of the one-shot optimization on the accuracy of the gradients is within acceptable bounds, the gradient is verified halfway the optimization process. The results are shown in fig. 7.2, at a point in the optimization where the objective function has decreased approximately two orders of magnitude. The discrepancy between the adjoint gradient and the gradient computed using finite differences is slightly higher than at the start of the one-shot optimization. This can be explained by the fact that the corrective term is changed every `adjointSimpleFoam` iteration, and thus the primal and adjoint equations are not fully converged. However, the gradients still agree fairly well and are considered sufficiently accurate for the optimization, as long as the optimization is started from converged primal and adjoint fields and the

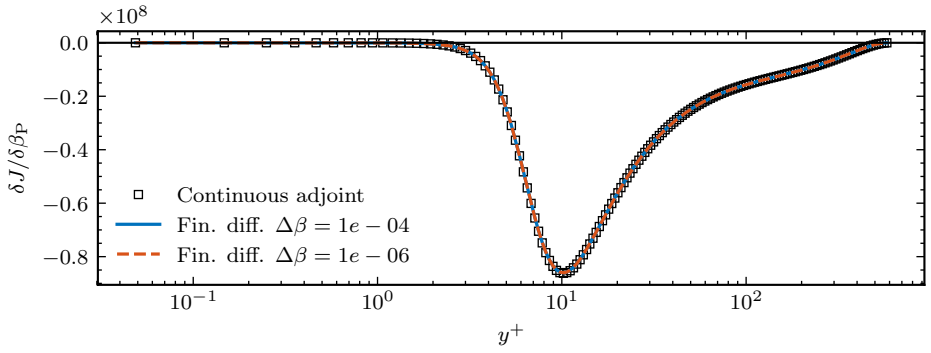


Figure 7.1: Gradient verification, turbulent channel flow ($\mathrm{Re}_\tau = 590$).

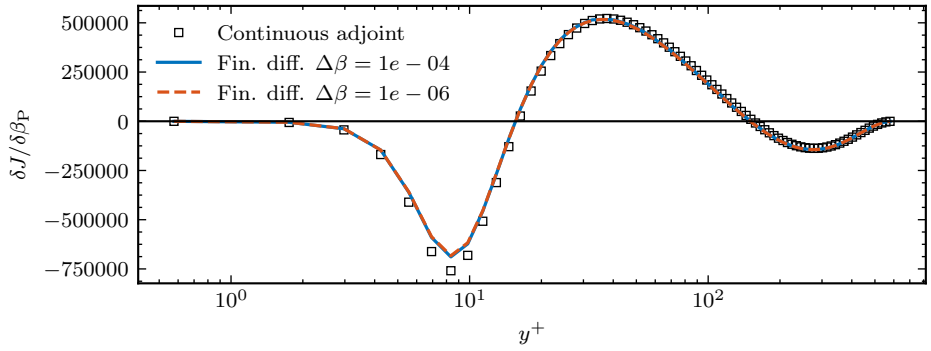


Figure 7.2: Gradient verification, turbulent channel flow ($Re_\tau = 590$) - during one-shot optimization.

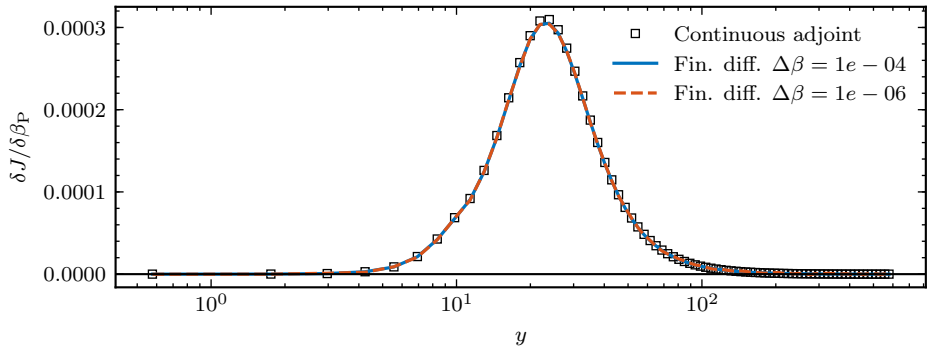


Figure 7.3: Gradient verification, turbulent channel flow ($Re_\tau = 590$), Hessian approximation objective function.

gradient descent step size is sufficiently low. Again, the finite difference perturbation size has no visible effect on the resulting gradient.

The adjoint solver used for the approximation of the Hessian uses a different objective function and thus a different source term in the adjoint momentum equation. Therefore, the gradients are also verified for this version of the solver. The resulting gradients are shown for the same flow case and the objective function corresponding to $i = 7$ in fig. 7.3, using (6.26). The corrective term is equal to one for all y. Again, both methods yield practically the same result and thus the implementation of the gradients used for the Hessian approximation can be considered verified for this flow case.

In order to verify the correct implementation in the adjoint solver on a more complex case, the resulting adjoint gradient is verified with finite differences for the periodic hills flow case at Re = 5,600. The gradient was verified at multiple streamwise locations. The gradient is not verified for the full domain because of the computational cost involved with the finite difference calculation, especially when this is performed for multiple perturbation sizes. For the sake of brevity, only the results for one streamwise location and one flow case are shown here. The adjoint gradient for the full domain is shown in fig. 7.4, indicating the line along which the results of the gradient verification are shown. The comparison between the adjoint gradient and the gradient computed using finite differences is shown in fig. 7.5. The gradient is zero in most of the domain, in areas where the $k-\omega$ model provides satisfactory results or where the production term of the ω-equation is not effective in matching the streamwise mean velocities. The gradients calculated using the two different methods match fairly well, although a larger discrepancy is observed compared to the verification of the gradient in the turbulent channel flow case. The discrepancy is of comparable magnitude as for the continuous adjoints verified in previous works such as Schramm et al. [130], Zymaris et al. [172], or Kavvadias et al. [71] for flow cases of similar complexity. It can therefore be expected that it is not caused by an error in the implementation but that it is inherent to the use of the continuous adjoint. Nevertheless, the adjoint gradient is considered to be accurate enough for the purpose of the optimization in the field inversion phase and has been found empirically to perform well, as long as the adjoint equations are sufficiently converged throughout the optimization. Similar results were obtained for the gradient verification at other streamwise

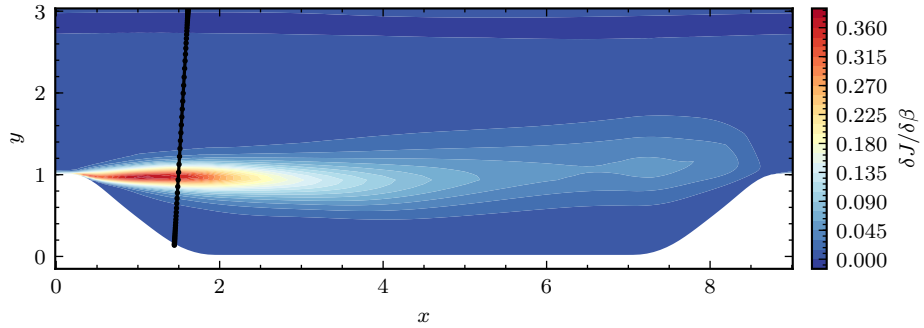


Figure 7.4: Adjoint gradient at $\beta_P = 1$, indicating the line along which the gradient is verified, periodic hills (Re = 5,600).

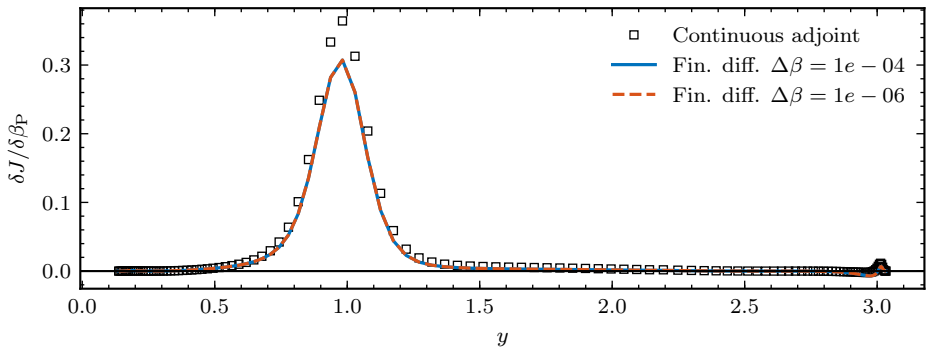


Figure 7.5: Gradient verification, periodic hills (Re = 5,600), locations indicated in fig. 7.4.

locations and other flow cases.

7.1.2. Comparison between the one-shot and complete approach

As described in section 6.4, a study was performed in order to investigate the effect of the step size on the optimization using the one-shot approach and the effect of the convergence criteria on the optimization using the complete approach. The results are shown in fig. 7.6. All simulations start from converged primal and adjoint fields for $\beta_P = 1$. For the one-shot approach, it can indeed be seen that selecting a small step size slows down convergence. On the other hand, a high step size results in a rapid decrease in the objective function in the start of the optimization process, but diverges once the gradients become increasingly inaccurate. For optimal convergence, it is necessary to find an intermediate value, trading off these effects.

For the complete approach, the markers indicate the `adjointSimpleFoam` iterations at which a new corrective term was introduced by the optimization algorithm. The conjugate gradients (CG) optimization method is used for the complete approach. A new corrective term is given by the optimization algorithm once all primal and adjoint equations are converged below the indicated tolerance. As expected, a more lenient convergence criterion improves the convergence of the optimization process. However, similarly to selecting a high step size, the resulting gradients are less accurate. Requiring a further converged result, on the other hand, results in more accurate gradients but slows down the optimization process. Overall, it can be seen that carefully chosen intermediate values yield similar performance for both methods. However, the one-shot approach tends to decrease the objective function more reliably once the (local) minimum is approached. Surprisingly, as long as the optimization was started from converged primal and adjoint fields and an accurate value for the step size was chosen, the one-shot approach was found to be more robust than the complete approach in all flow cases. It would be expected that the complete approach would be more robust, as one diverging simulation does not necessary lead to divergence of the full optimization, as is the case for the one-shot approach. Furthermore, the complete approach demanded similar requirements on the convergence of the primal fields for convergence.

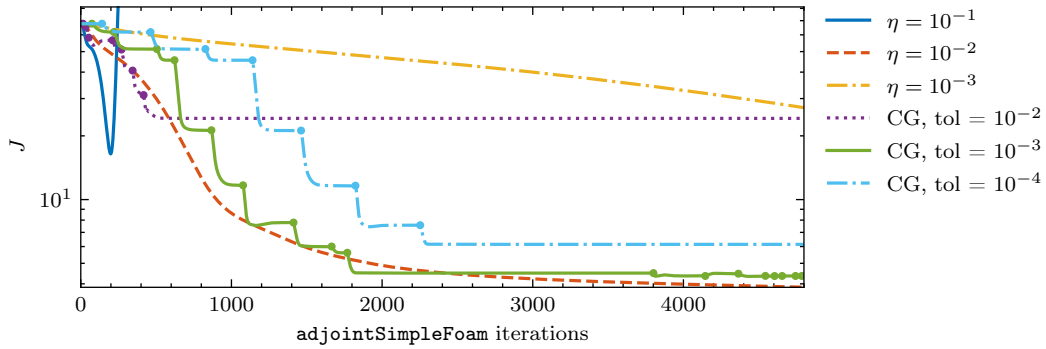


Figure 7.6: Effect of the gradient descent step size and convergence criteria on the convergence of the optimization process, converging-diverging channel (Re = 12,600), CG = conjugate gradients.

7.2. Maximum a posteriori results

Now the gradients resulting from the adjoint solver are verified and the behavior of the two optimization methods is investigated, the results of the field inversion phase will be presented for each flow case. Per case, the results of the correction to the production term in the ω-equation are presented first, after which the results of correcting the eigenvalues are shown. The correction to the production term in the ω-equation will from now on be referred to as the production term correction.

7.2.1. Turbulent channel flow

The simplest flow case considered in this work is turbulent channel flow, which was introduced in section 5.1. The $k-\omega$ model is relatively accurate in predicting the mean velocity profiles in this flow. However, as data is available for several Reynolds numbers it is still interesting to see how the inferred corrections scale with the Reynolds number. Furthermore, because of the low computational cost, the posterior covariance can be shown in addition to the mean inferred results.

The inferred velocity profile using the production term correction for $\mathrm{Re}_\tau = 395$ is shown in fig. 7.7. There is a strong agreement between the MAP solution and the data, showing that the production term correction has correctly inferred the mean velocity at all wall distances. The corresponding corrective term is shown in fig. 7.8, indicating the $\pm 2\sigma$ limits, where the standard deviation is obtained using the approximate Hessian. In the viscous sublayer ($y^+ < 5$), turbulent production is low, and thus the corrective term is not effective in this region. This is reflected well by the posterior distribution: the MAP of the corrective term collapses to the value corresponding to the base model ($\beta_\mathrm{P} = 1$), and the posterior variance is high, matching that of the prior distribution. Due to the low observational covariance, the mean velocity further away from the wall is inferred accurately and the uncertainty is low. A similar pattern can be observed for all Reynolds numbers considered, as can be seen in fig. 7.9. It is interesting to see that there is a clear scaling with the Reynolds number, especially for the higher Reynolds number cases. This scaling is a desirable property for a turbulence model to have and is often absent in turbulence models [107]. These kind of results might yield valuable information for the development of new turbulence models. Investigating the eddy viscosity might provide further insight in the resulting shape of the corrective function. The equivalent eddy viscosity can be extracted from the DNS data using

$$\nu_{t,\mathrm{DNS}} = -\frac{\overline{v_1' v_2'}}{\partial v_1 / \partial x_2} \tag{7.1}$$

The resulting profiles are shown in fig. 7.10. The eddy viscosity corresponding to the MAP solution and the DNS data agree fairly well. Furthermore, multiple regions can be identified. Close to the wall, the eddy viscosity predicted by the $k-\omega$ model is practically equal to the DNS data. In a region roughly corresponding to the viscous sublayer, between $1 < y^+ < 5.5$, the $k-\omega$ model underpredicts the eddy viscosity slightly, although the difference is too small to be observed in the figure. Moving further away from the wall, this is followed by three regions in which the $k-\omega$ model shows an alternating pattern of overprediction and underprediction of the eddy viscosity. These four regions in total roughly correspond to the four alternating peaks in the corrective term in fig. 7.9. In regions where the $k-\omega$ model underpredicts the eddy viscosity, the corrective term is slightly lower than one, and vice versa. This is consistent with the fact that the corrective term is applied to

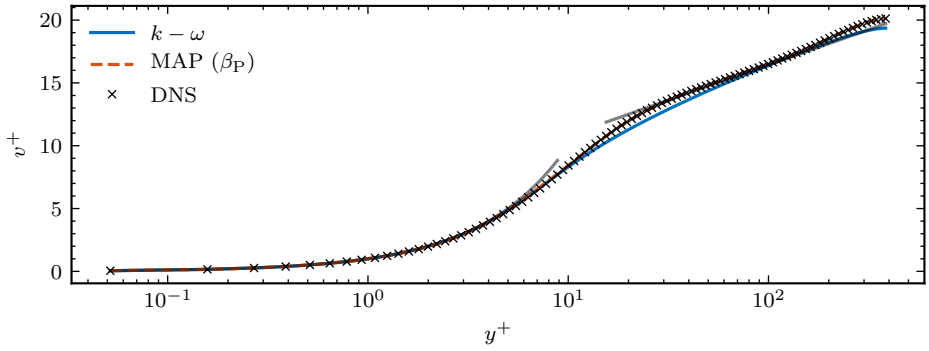


Figure 7.7: Velocity profiles using inferred β_P, turbulent channel flow ($Re_\tau = 395$).

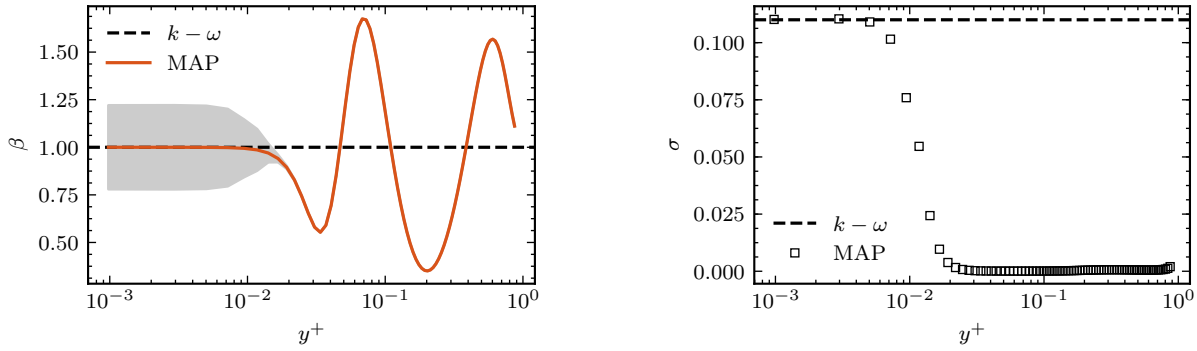


Figure 7.8: Inferred corrective term $\beta_{MAP} \pm 2\sigma$, turbulent channel flow ($Re_\tau = 180$), with σ obtained using the approximate Hessian.

the production term in the equation for ω, which appears in the denominator of the expression for the eddy viscosity.

The mean streamwise velocity profile resulting from the eigenvalue corrections is shown in fig. 7.11. The inferred mean velocity shows a comparable agreement with the DNS data as for the production term correction. The inferred correction shows an improvement over the $k-\omega$ model in nearly all regions. The resulting paths in the barycentric map are shown in fig. 7.12. Starting with the DNS data, it can be seen that there is two-component turbulence near the wall. This can be derived by expanding the fluctuating velocities near the wall. Following Pope [116, p. 283], for small y and for fixed x, z, and t, the following can be written:

$$\begin{aligned} v_1' &= a_1 + b_1 y + c_1 y^2 + \ldots, \\ v_2' &= a_2 + b_2 y + c_2 y^2 + \ldots, \\ v_3' &= a_3 + b_3 y + c_3 y^2 + \ldots \end{aligned} \tag{7.2}$$

The no-slip and impermeability conditions require that $a_1 = a_2 = a_3 = 0$. Another consequence of the no-slip condition is that $(\partial u/\partial x)_{y=0}$ and $(\partial w/\partial z)_{y=0}$. Furthermore, from the continuity equation it follows that $b_2 = 0$. Approaching the channel centerline, the axisymmetric condition ($b_{22} = b_{33}$) is met. The flow is still anisotropic at this location, though to a lesser degree than in the log-law region [8]. As expected, the eigenvalues predicted by the $k-\omega$ model all lie along the plane strain line. The inferred corrections show a considerable deviation from this plane-strain behavior. However, it can be seen that the inferred eigenvalues are completely different than the eigenvalues corresponding to the DNS data. The MAP eigenvalues near the wall correspond to isotropic flow. Moving away from the wall, the axisymmetric contraction line is followed. Near the centerline, the path moves towards the middle of the barycentric map. This demonstrates an important point: even though the quantity of interest has been inferred correctly, the corrective term does not correspond to the ground truth. The reason behind this finding and its implication for (data-driven) turbulence modeling will be further discussed at the end of this section.

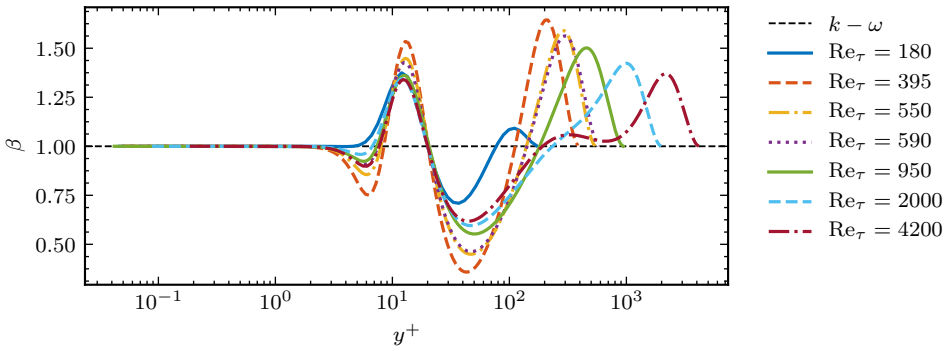


Figure 7.9: Inferred corrective terms for turbulent channel flow, all Reynolds numbers.

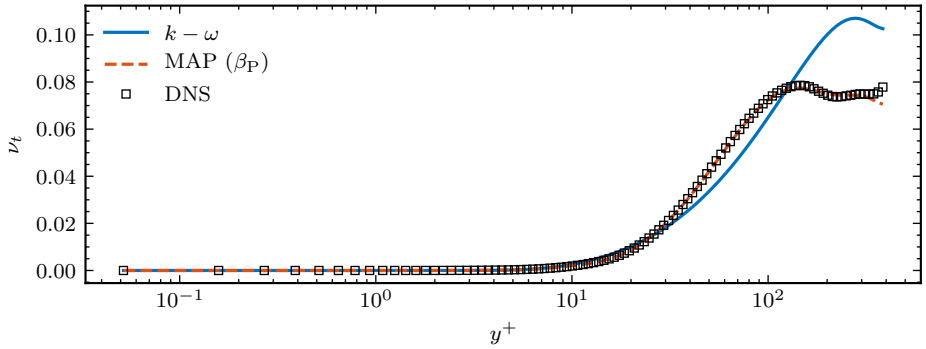


Figure 7.10: Eddy viscosity, turbulent channel flow ($\mathrm{Re}_\tau = 395$).

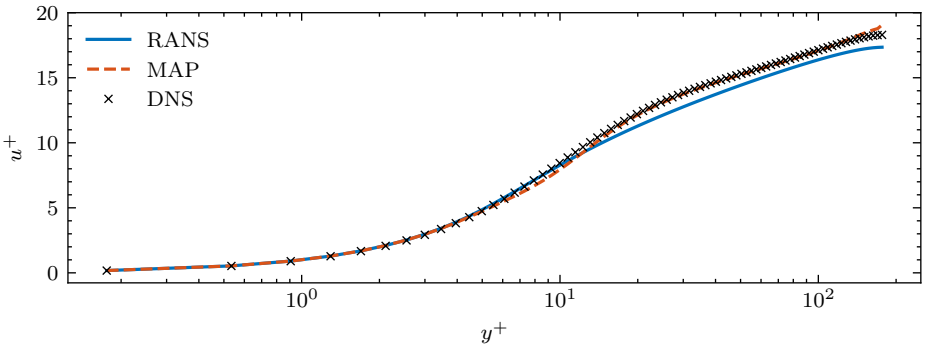
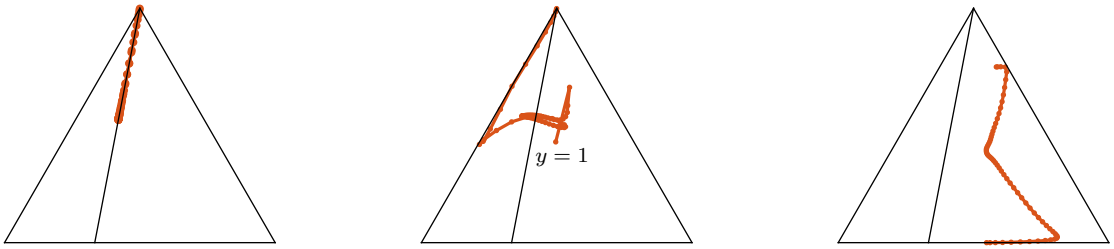


Figure 7.11: Velocity profile using inferred $\beta_{C_{1c}}$ and $\beta_{C_{2c}}$ ($\mathrm{Re}_\tau = 180$).



(a) Baseline $k-\omega$ model (b) MAP solution (c) DNS data

Figure 7.12: Paths in the barycentric map, turbulent channel flow ($\mathrm{Re}_\tau = 180$).

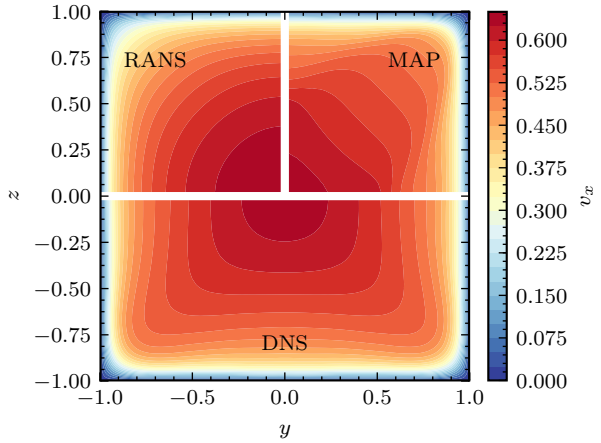


Figure 7.13: Streamwise velocity, square duct (Re = 2,400).

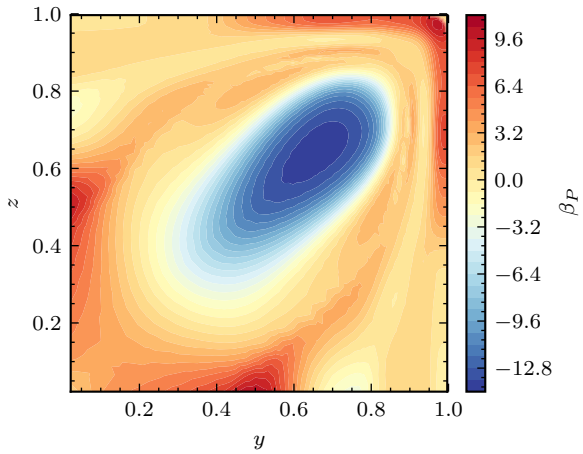


Figure 7.14: Corrective term, square duct (Re = 2,400).

7.2.2. Square duct

The first two-dimensional flow case considered in this work is the square duct flow. As described in section 5.2, the main phenomenon of interest are the secondary motions. These motions are absent in linear closure, as they originate entirely in turbulence anisotropy. The objective function used for generating the results described for the production term correction only includes the discrepancy in the streamwise velocity, as the in-plane velocities for the corrected model will be zero due to the Boussinesq assumption, independent of the correction to the production term. The streamwise velocities for the base model, the MAP solution, and the DNS data are shown in fig. 7.13 for Re = 2,400. The MAP solution shows a qualitative improvement in the distribution of the streamwise velocity with respect to the velocities predicted by the $k-\omega$ model. The most notable improvement is the increased streamwise velocity in the corner of the channel. The secondary flows, which cause an increased momentum transfer to the corners, are absent in the baseline RANS model, resulting in an underprediction of the streamwise velocity in the corner [91]. The corresponding streamwise velocities can be found fig. 7.15, along various horizontal lines located at $z = 0.1, 0.3, 0.7, 0.9$. Here, it can be seen that the field inversion infers a corrective term which results in a streamwise velocity which closely matches that of the high-fidelity data at almost all locations. A small discrepancy between the MAP and the DNS results can be observed near the center of the channel ($z = 0.1$), around $y = 0.5$. At this location, the streamwise velocity predicted by the $k-\omega$ model is actually closer to the DNS data than the MAP prediction. This might be caused by the optimization being stuck in a local minimum. Another explanation might be that better matching the streamwise velocity in another location forces the corrective term to a value which causes a deviation from the DNS data in this location. To some extent, this behavior is inherent to optimizing an integral quantity by varying a high-dimensional variable.

The corresponding corrective term is shown in fig. 7.14. Its value deviates from that of the base model in a large part of the domain. This can be expected given the deficiencies of the base model in modeling the underlying physics, which in this flow case affect the complete domain. This, in contrast to other geometries considered in this work, such as periodic hills or the converging-diverging channel, where the RANS model yields accurate streamwise velocities in large parts of the domain. This demonstrates an important aspect of the paradigm: if the corrective term is not formulated in a manner which reflects the missing physics in the base model, it is possible that the inferred corrective term does not yield any useful information. Surprisingly, the relatively high absolute values of the production term caused no issues with numerical stability, as might be expected in regions where the ω is pushed to high values and thus results in a low eddy viscosity. A summary of the resulting corrective terms for all Reynolds numbers is shown in fig. 7.16. No universal behavior is observed with respect to the Reynolds number. Again, this can be attributed to the non-physical nature of the corrective term in this specific flow case. It was found that the absolute value of the corrective term would slowly keep increasing the longer the optimization was continued. Even with a lower prior covariance, and thus a stronger regularization, this behavior was present. It is important to note that the behavior described thus far for this flow case is caused by the fact that this corrective term formulation is still limited by the Boussinesq assumption. Therefore, the in-plane velocities predicted for the MAP solution will always be

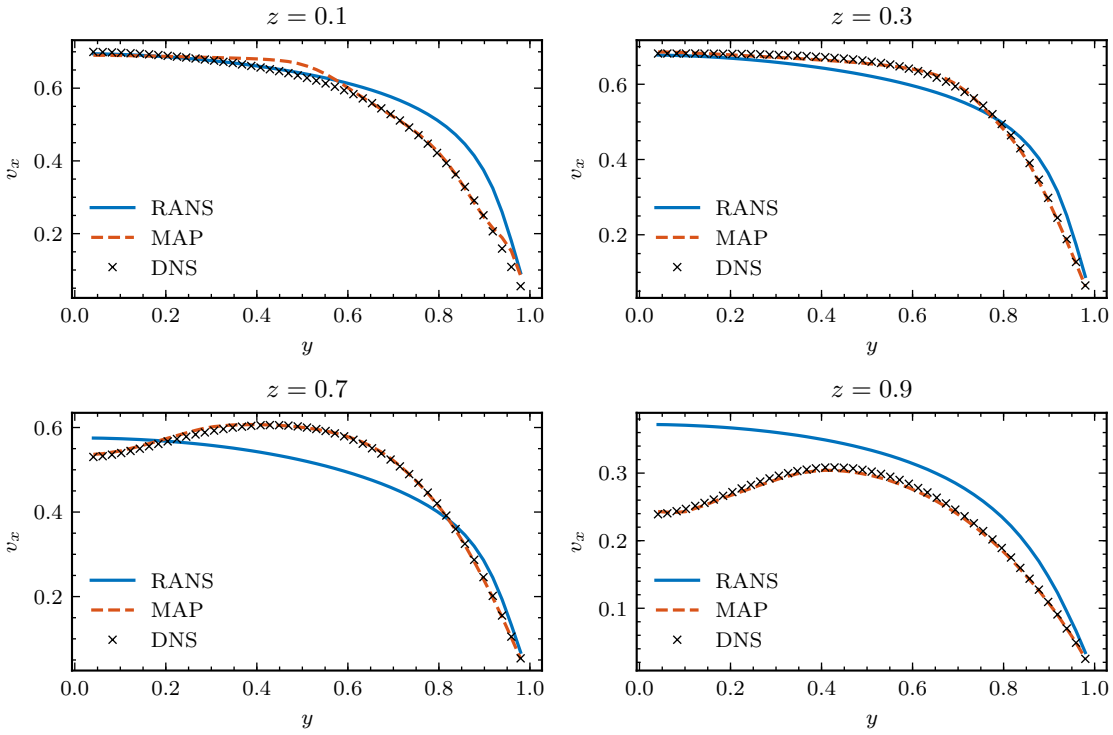


Figure 7.15: Streamwise velocity profiles along various spanwise locations, square duct (Re = 1,100).

zero. A corrective term inferring all three velocity components correctly is thus non-existent in this formulation. This is one example where the improved generality of formulating the corrective term as a correction to the eigenvalues is beneficial.

For the production term correction, only the discrepancy in the streamwise velocity was included in the objective function. In order to see whether the corrections to the eigenvalues are able to infer the in-plane velocities, these are now also included in the objective function. However, simply including the in-plane velocity components in the objective function did not yield accurate results for the inferred in-plane velocities. This is to be expected, given the relative magnitudes of the velocity components. The streamwise velocity is approximately two orders of magnitude larger than the in-plane velocities. Therefore, for the results presented in this section for the eigenvalue correction, the terms including the discrepancies for each velocity component were weighted according to the mean absolute values of each velocity component in the data. The in-plane velocities can be seen in fig. 7.17. In contrast to the production term correction, the correction to the eigenvalues results in non-zero in-plane velocities. However, the inferred velocity field is asymmetric

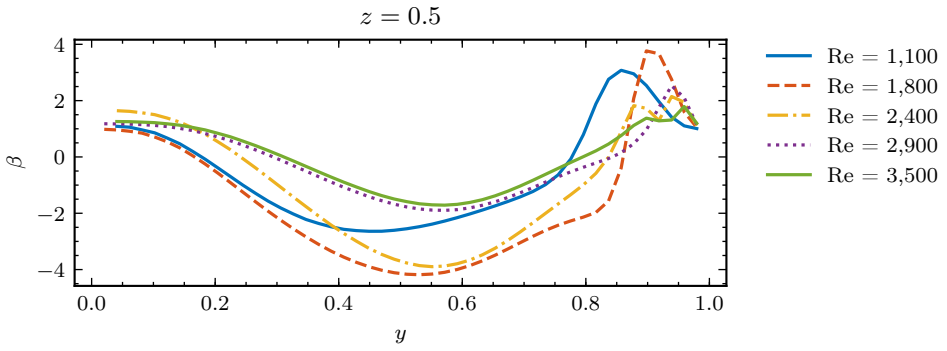


Figure 7.16: Inferred production term corrections for various Reynolds numbers.

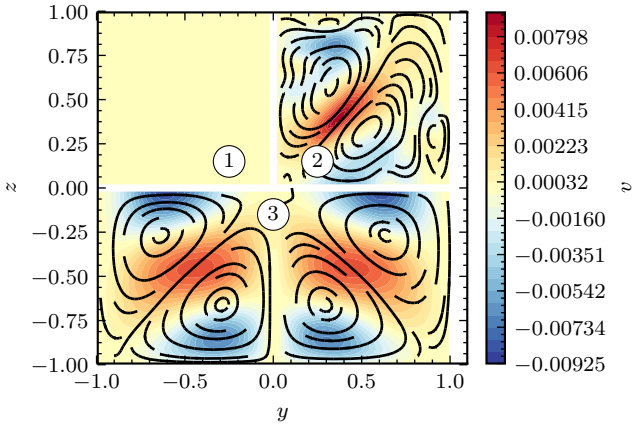


Figure 7.17: Spanwise velocity profiles, square duct (Re = 1,100), (1) = baseline $k-\omega$ model, (2) = MAP solution, (3) = DNS data.

and actually shows another small circulating region near the wall. The velocity profiles in fig. 7.18 also show that the in-plane velocities are inferred rather poorly.

In order to obtain more insight into what corrections to the eigenvalues the method is inferring, the eigenvalues of the anisotropy tensor are visualized using the barycentric map in fig. 7.19. As a result of the Boussinesq assumption, all eigenvalues from the $k-\omega$ flowfield lie on the plane-strain line. The eigenvalues corresponding to the MAP result, shown in fig. 7.19b, deviate from the plane-strain in most locations. However, the inferred eigenvalues differ significantly from eigenvalues from the DNS data, shown in fig. 7.19c. For this flow case, mainly one-component and three-component turbulence is present. Close to the wall, the fluctuations in the streamwise direction case one-component turbulence. Near the center of the channel, the influence of the walls is lower, and the turbulence is closer to the three-component limit. Again, the vast difference between the inferred eigenvalues and those corresponding to the data will be discussed at the end of this section.

7.2.3. Periodic hills

The third flow case for which the results will be presented is the periodic hills. This case differs from the previous two cases in that the $k-\omega$ model is accurate in large regions of the domain. Furthermore, the flow case features a large separated region, which is often inaccurately predicted by linear closures.

The optimized production corrective term for the periodic hills is shown in fig. 7.20 for Re = 5,600. The correction to the production term is mainly active in the shear layer, where it is slightly lowered with respect to the base value of one. The absolute values of the corrective term are significantly lower compared to those of the square duct flow case. Contrary to the previous two flow cases, the corrective term corresponds to the base model in most of the domain. This can partly be explained by the fact that the $k-\omega$ model provides satisfactory results in terms of the streamwise velocity in these regions, in contrast to the square duct flow case, where the absence of the secondary flows in the $k-\omega$ model affects the full domain. In order to see the effect the corrective function has on the resulting eddy viscosity, the difference between the eddy viscosity predicted by the $k-\omega$ model and that of the MAP result is shown in fig. 7.21. The eddy viscosity is considerably higher in the downwind part of the shear layer. This is the same effect as was targeted by the ad-hoc fix by Rumsey [129], in an attempt to increase mixing and thus attain an earlier reattachment of the separated flow. Furthermore, the fact that the eddy viscosity of the MAP solution is higher than that of the base model is consistent with the observation that the inferred corrective term is lower than or equal to one.

Before looking at the resulting velocity field and reattachment location, it is first interesting to shortly discuss the turbulent kinetic energy. RANS models in periodic hill flow commonly underpredict the turbulent kinetic energy in the shear layer [129]. The turbulent kinetic energy profiles are shown at various streamwise locations in fig. 7.22. First of all, it can indeed be seen that the $k-\omega$ model underpredicts the turbulent kinetic energy. This is also the case for the MAP solutions, although to a lesser extent, especially closer to the downstream hill. It is also interesting to note that both formulations of the corrective term result in a similar profile for the turbulent kinetic energy. The resulting streamwise velocity profiles are shown in fig. 7.23. The

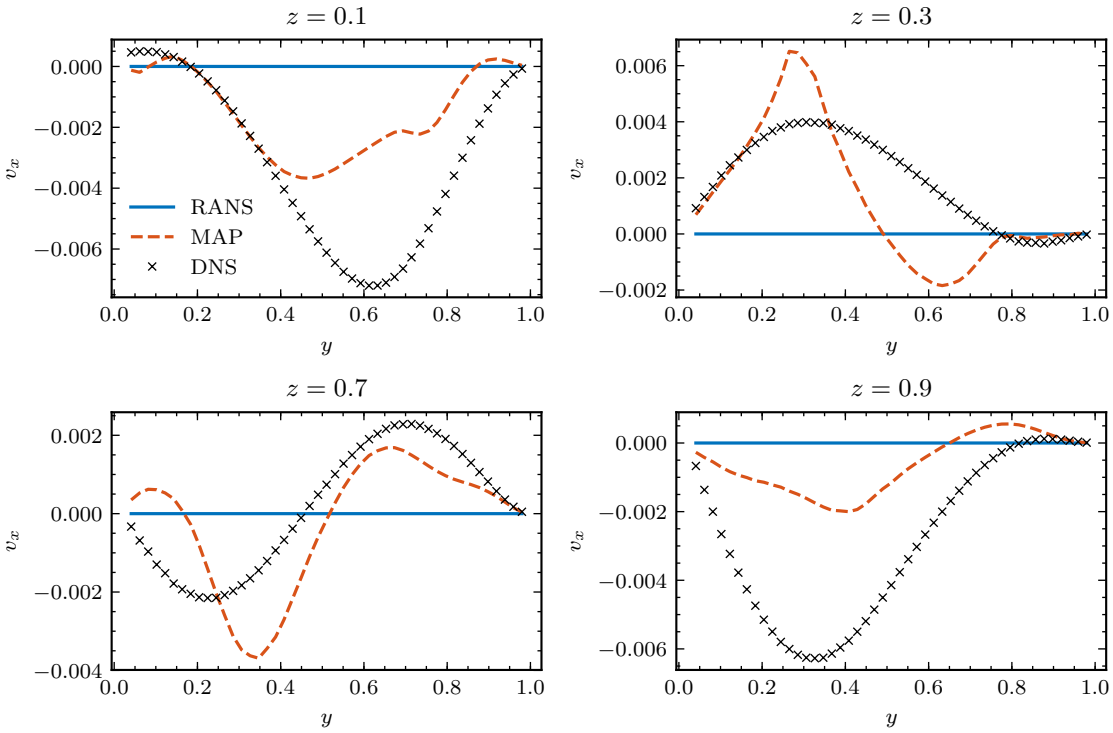
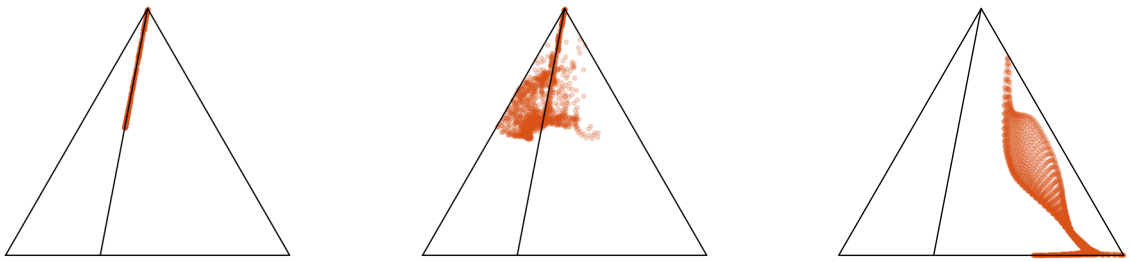


Figure 7.18: Spanwise velocity profiles along various spanwise locations, square duct (Re = 1,100).



(a) Baseline $k-\omega$ model (b) MAP solution (c) DNS data

Figure 7.19: Barycentric maps, square duct (Re = 1,100).

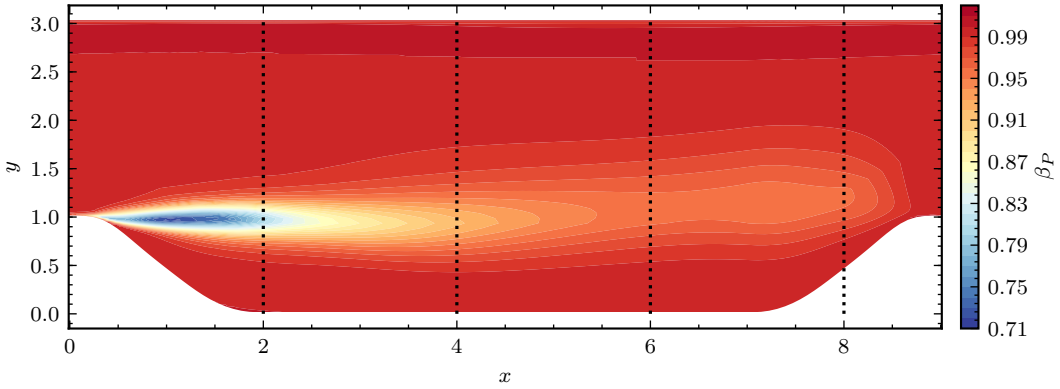


Figure 7.20: Inferred corrective function, periodic hills (Re = 5,600).

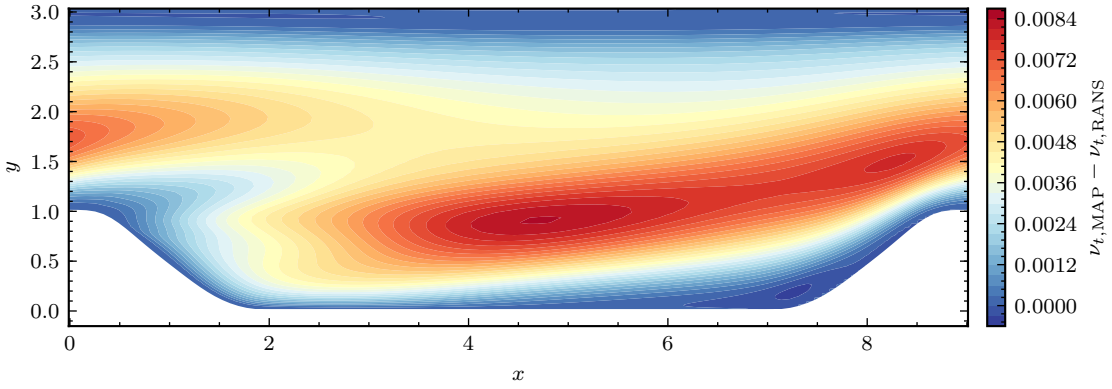


Figure 7.21: Difference between eddy viscosity predicted by $k-\omega$ and the inferred eddy viscosity, periodic hills (Re = 5,600).

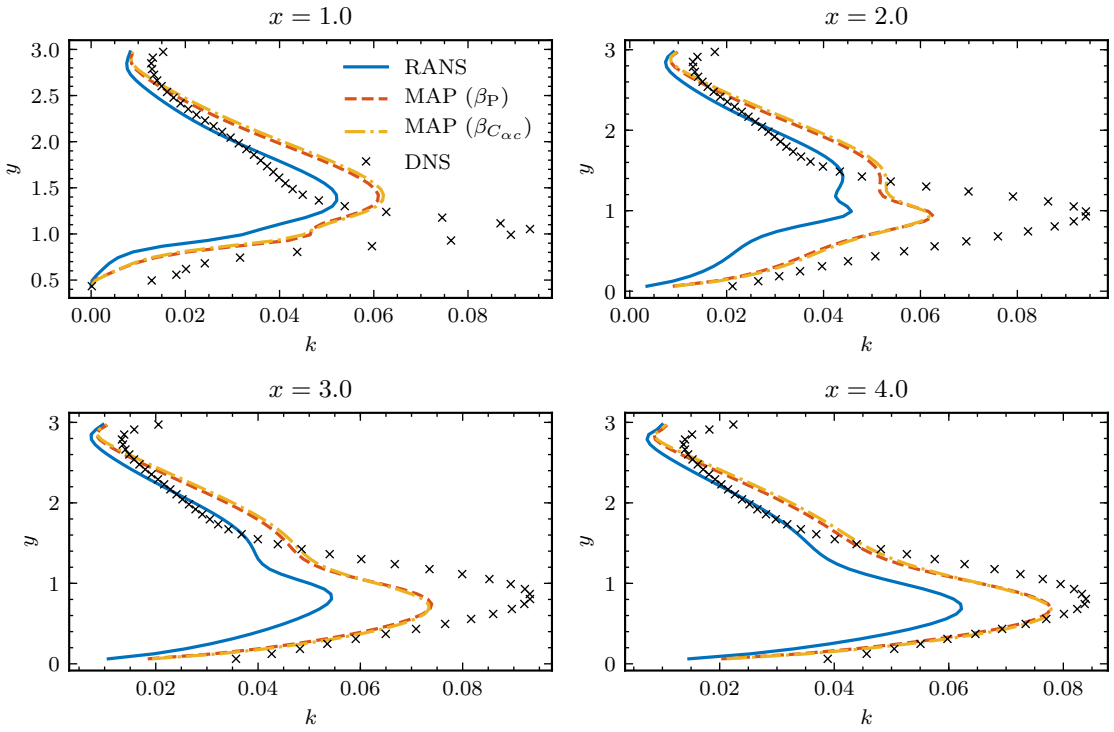


Figure 7.22: Turbulent kinetic energy at various streamwise locations, periodic hills (Re = 5,600).

MAP solutions provide an improvement over the streamwise velocities predicted by the $k-\omega$ model in most locations, especially near the walls. Again, there is no notable difference between the two formulations of the corrective term. In the regions where the corrective production term is equal to one, the RANS model already resulted in accurate streamwise velocities. Finally, the main quantity of interest for this flow case will be discussed: the reattachment location. The wall shear stress is shown in fig. 7.24. For the DNS data, the flow separates at $x = 0.18$ and reattaches at $x = 5.09$ [15]. The $k-\omega$ model is relatively accurate in predicting the separation location. However, the small discrepancy which exists is not corrected in the inferred results. For the reattachment location, the increased eddy viscosity in the shear layer observed in fig. 7.21 achieves the desired effect, as the MAP result of the production term correction nearly matches the reattachment location of the high-fidelity data. The inferred wall shear stresses of the two corrective term formulations agree fairly well.

The corresponding visualizations of the eigenvalues using the RGB-colormap are shown in fig. 7.25. Starting with the DNS data in fig. 7.25c, it can be seen that throughout the domain all limiting states are present. The center of the channel features isotropic turbulence. At the bottom wall, there is mainly two-component turbulence whereas the upper wall and the windward side of the downstream hill feature one component turbulence. The latter can be explained by the splatting effect [39]: eddies originating from the shear layer are convected to the downstream hill where they cause large fluctuations in the z-direction. The stress types predicted by the $k-\omega$ model, visualized in fig. 7.25a, are significantly different. This is due to the anisotropy present in the flow case which cannot be represented by the RANS model. For example, the b_{33} component of the anisotropy tensor will always be zero, as in the linear model it will be related to $\partial v_1/\partial x_3$ and $\partial v_3/\partial x_1$, which are both zero due to homogeneity in the z-direction.

7.2.4. Converging-diverging channel

The final geometry considered in this work is the converging-diverging channel. Similarly to the periodic hills, the $k-\omega$ model is relatively accurate for large parts of the domain. However, the separated region is much smaller. Nevertheless, the baseline $k-\omega$ model does not predict this region accurately in terms of its location and its length.

The inferred production correction for the converging-diverging channel is shown in fig. 7.26, at a Reynolds number of Re = 12,600. Similarly to the field inversion results of the periodic hills flow case, the production

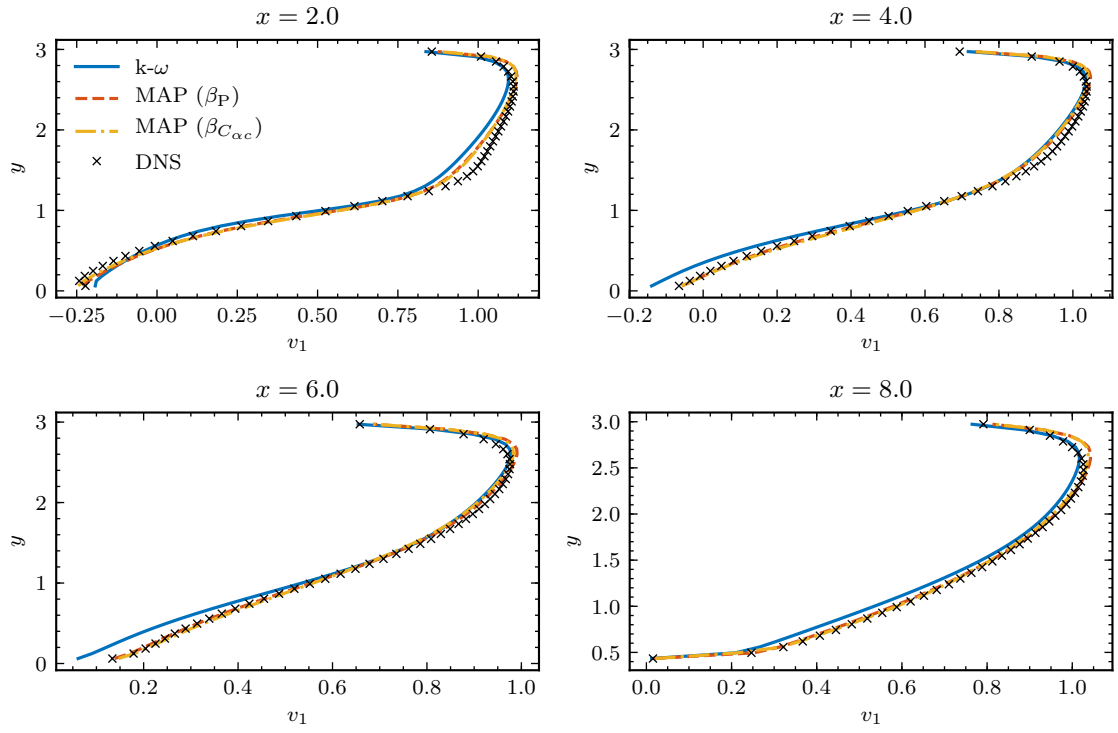


Figure 7.23: Inferred velocity profiles at various streamwise locations, periodic hills (Re = 5,600).

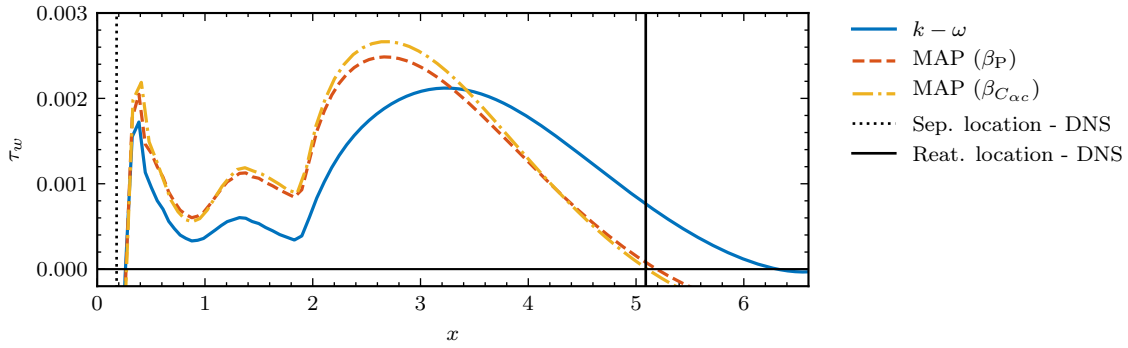
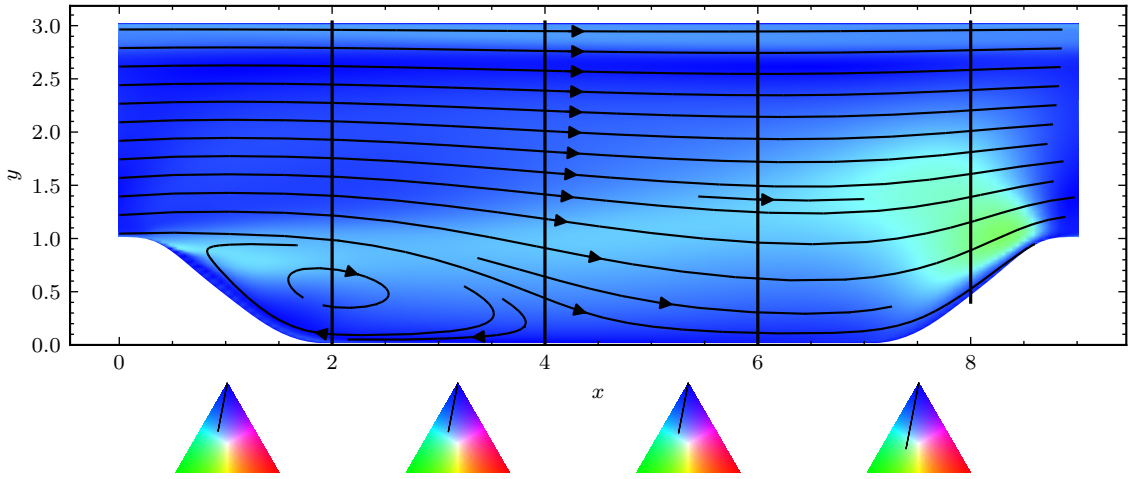
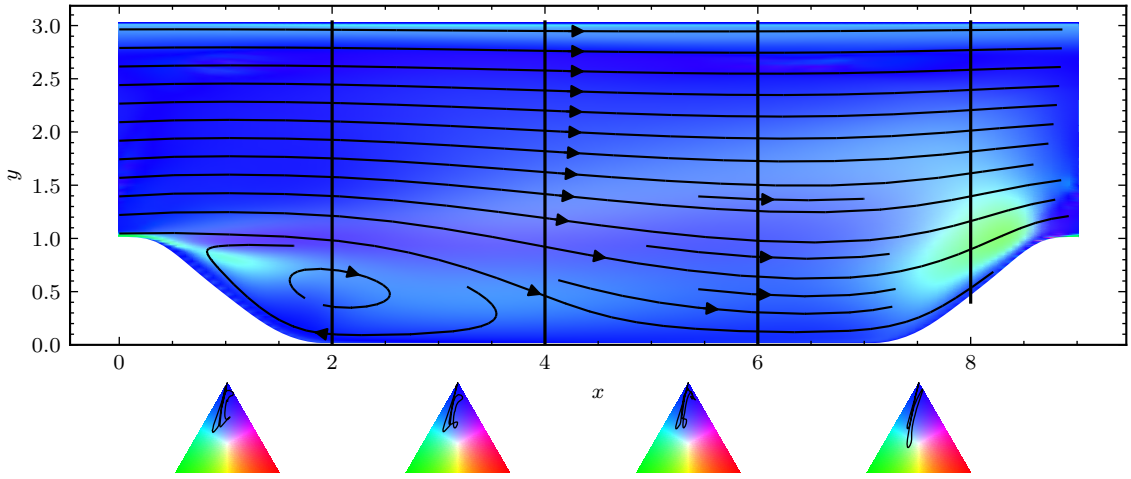


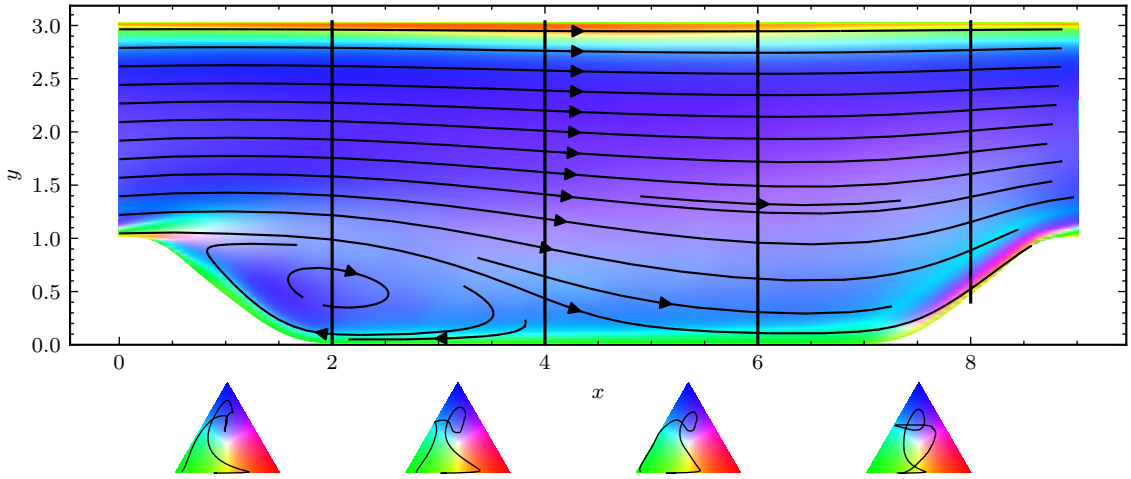
Figure 7.24: Wall shear stress on the bottom wall of the periodic hills (Re = 5,600).



(a) $k-\omega$ base model



(b) MAP solution



(c) DNS data

Figure 7.25: Eigenvalue visualizations, periodic hills (Re = 5,600).

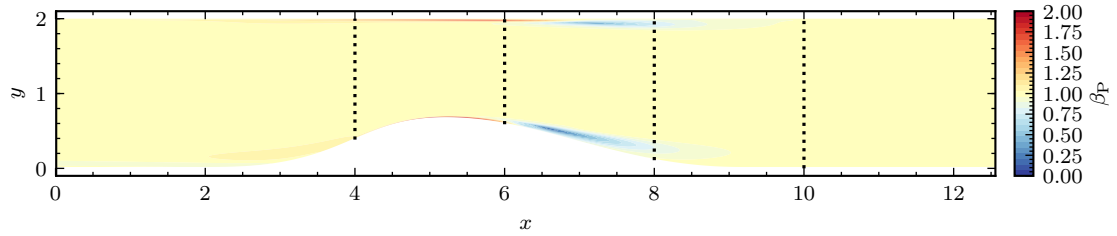


Figure 7.26: Inferred β_P, converging-diverging channel (Re = 12,600).

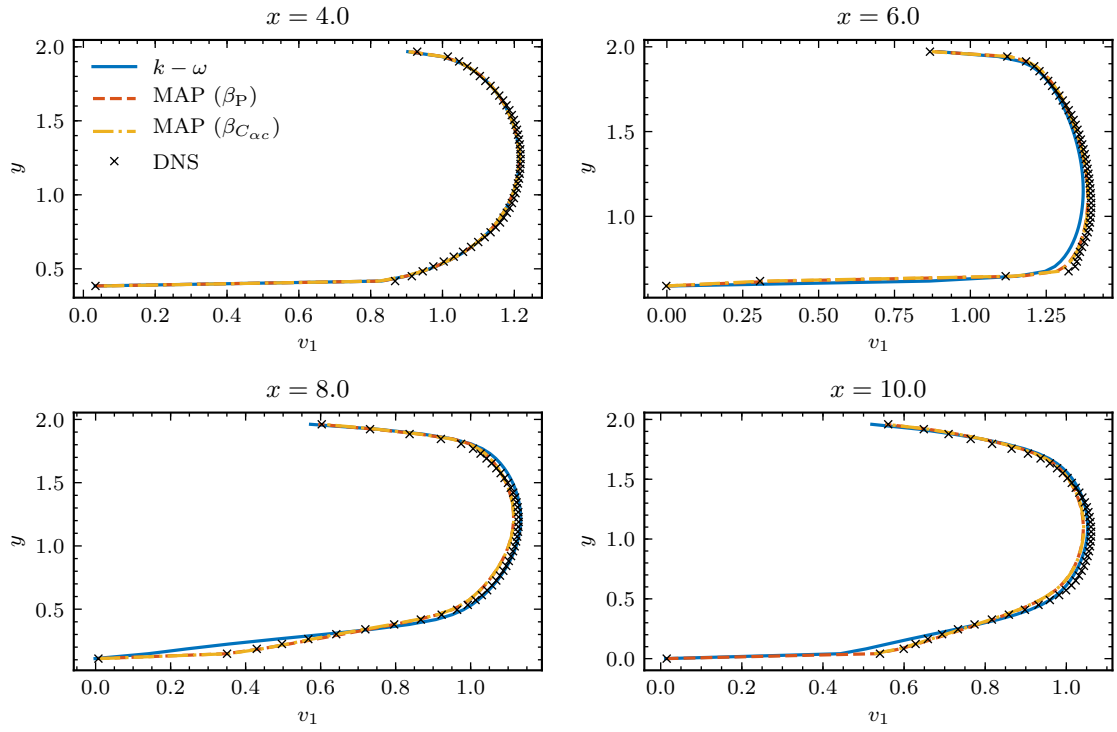


Figure 7.27: Inferred velocity profiles at various streamwise locations, converging-diverging channel (Re = 12,600).

term correction is unity in most of the domain, reducing the augmented model to the uncorrected $k-\omega$ model. Around the narrowest part of the channel, the production term is slightly increased near the walls, whereas the production term is decreased slightly downstream of these locations. It might be hypothesized that this increase in the corrective term is necessary for correcting the separation location. In this flow case, the prediction of the separation location is less accurate than for the periodic hills, where the increase in the corrective term was not observed. The lines along which the velocity profiles will be shown are indicated. Again, this shows that in regions where the base model is sufficiently accurate in terms of the quantities of interest, no correction is applied. The corresponding streamwise velocity profiles are shown in fig. 7.27. First of all, it should be noted that the $k-\omega$ model is quite accurate in predicting the streamwise velocities at most locations. Only at the leeward side (around $x = 8.0$) the velocity near the bottom wall is inaccurate for the RANS model. Again, at this location, the streamwise velocity is correctly inferred by both corrective term formulations.

The spatial variation of the stress types are displayed using the RGB-colormap in fig. 7.29. Again, the DNS data, shown in fig. 7.29c, shows a large amount of anisotropy, which is not predicted by the base model. As expected, there is mainly three-component turbulence in the center of the channel. On a large part of the upper wall and in the separated region on the hill, one-component turbulence is present. In the region with a favorable pressure gradient (approximately $3 < x < 5$) and downwind of the hill, two-component turbulence can be observed. The stress types corresponding to the inferred eigenvalues are shown in fig. 7.29b.

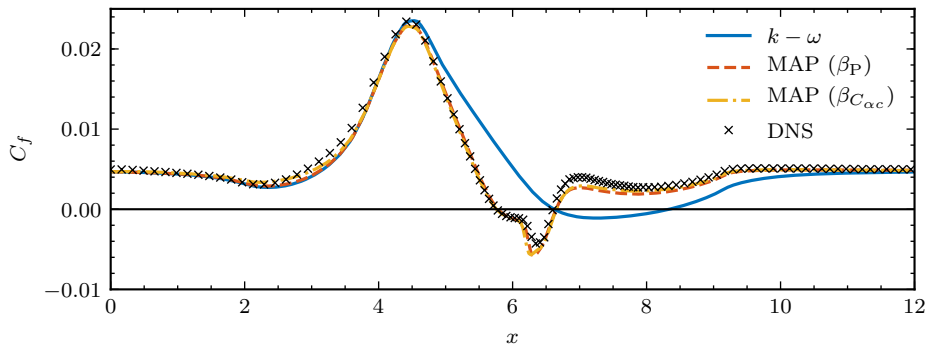
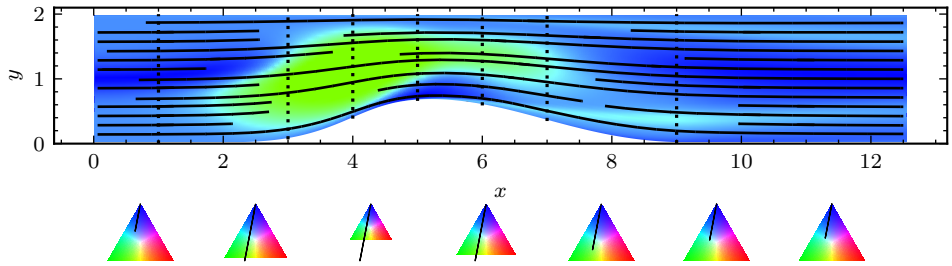


Figure 7.28: Skin-friction coefficient, converging-diverging channel (Re = 12,600).

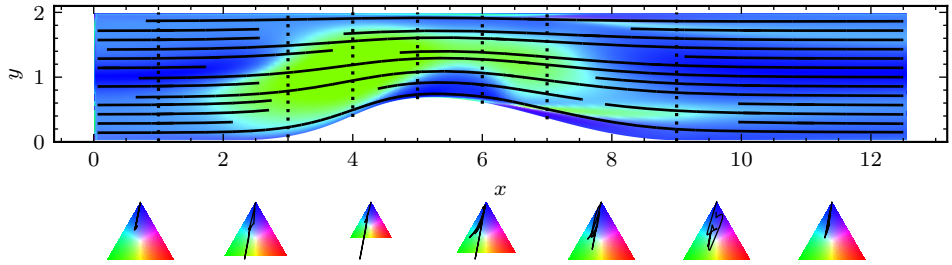
Upstream of the hill, the inferred stress types are practically the same as those predicted by the base model. On the windward side of the hill, unrealizable turbulence is predicted. In this region, the correction to the eigenvalues is also negligible. From the earlier observation that the streamwise velocities predicted by the $k-\omega$ model are close to those of the DNS data, this is not surprising. In the diverging part of the channel however, the MAP eigenvalues differ more from the RANS result. However, again, the resulting paths in the barycentric map do not resemble those of the DNS data.

For most flow cases considered in the this section, it was observed that both corrective term formulations were able to infer the mean velocities accurately. However, the inferred Reynolds stress fields did not resemble those corresponding to the high-fidelity data. This can be explained by the many-to-one mapping which exists from the Reynolds stress tensor to the mean velocities [169], i.e. a given mean velocity field can correspond to many different Reynolds stress fields. This can partly be attributed to the fact the Reynolds stress tensor enters the momentum equation through the divergence [169]. Furthermore, it makes intuitive sense that the mean velocity data only is insufficient to correctly infer the high-dimensional stress field. As the optimizations performed in the field inversion phase always started from the primal fields from the $k-\omega$ model, the optimizations converged to a local minimum close to these baseline results. The optimized eigenvalues are thus still close to those corresponding to the RANS results. This local minimum was sufficiently accurate in terms of the quantities of interest for most of the flow cases.

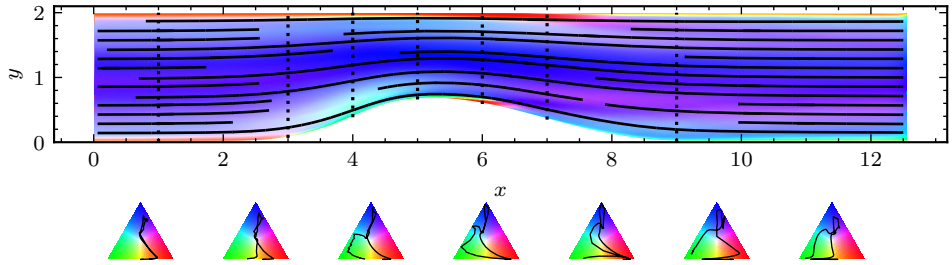
Wu et al. [167] defines the *model conditioning* as the sensitivity of the calculated fields resulting from solving the model (e.g. the mean velocities) to the modeled terms, in this case the Reynolds stress tensor. A natural question to ask next is what the impact is of the ill-conditioning of the RANS models using Reynolds stress closures. Ill-conditioning can cause the errors in the Reynolds stresses to be amplified in the velocity fields or other observed quantities. It can therefore be problematic for both data-driven and traditional approaches to modeling the Reynolds stress tensor. This was shown, for example, in Thompson et al. [149]. Accurate Reynolds stresses were propagated for a turbulent channel flow at various Reynolds numbers. It was found that the resulting mean velocity profiles show a considerable discrepancy compared to the DNS data at higher Reynolds numbers. These findings were confirmed by Poroseva et al. [118]. In the context of the paradigm of FIML, inferring the eigenvalues of the Reynolds stress tensor can still be considered as a feasible approach. There is an advantage in generality compared to inferring a correction to the production term in the ω-equation, where effectively only the magnitude of the Reynolds stress tensor is addressed. Furthermore, from the perspective of uncertainty quantification, the correction is introduced to the main source of uncertainty in the RANS equation [100]. The number of possible Reynolds stress fields may be further constrained by considering more restrictive prior distributions. However, it is unlikely that this will result in significant improvements in the inferred eigenvalues [167].



(a) $k-\omega$ base model



(b) MAP solution, eigenvalue corrections



(c) DNS data by Laval and Marquillie [76]

Figure 7.29: Eigenvalue visualizations, converging-diverging channel (Re = 12,600).

8

Results - machine learning

In the previous chapter, it was found that it was possible in most cases to accurately infer the quantities of interest using a correction to the base model. In general, it was found that these corrective functions are not always interpretable or unique. This chapter will investigate another desirable property of the corrective terms, namely their capabilities in generalizing to other Reynolds numbers, which is investigated in section 8.1, or flow cases, described in section 8.2, using machine learning algorithms. The effect of applying a second machine learning correction is shortly discussed in section 8.3. The chapter will be concluded by a brief discussion on the relative feature importances given by the random forests, in section section 8.4. For the results presented in this chapter, the production term correction is used.

Experiment	Training cases	Test case	Algorithm	$\frac{J_{\mathrm{ML}}-J_{\mathrm{RANS}}}{J_{\mathrm{RANS}}} \times 100$
I	TCF180-950, TCF4200	TCF2000	GP	−69.7%
II	SD1100, SD1800, SD2400, SD2900	SD3500	RF	−81.0%
III	PH5600, PH10595, CDC12600, TCF180-4200	BFS5100	RF	−48.6%
IV	CDC12600, TCF180-4200	BFS5100	RF	−13.1%

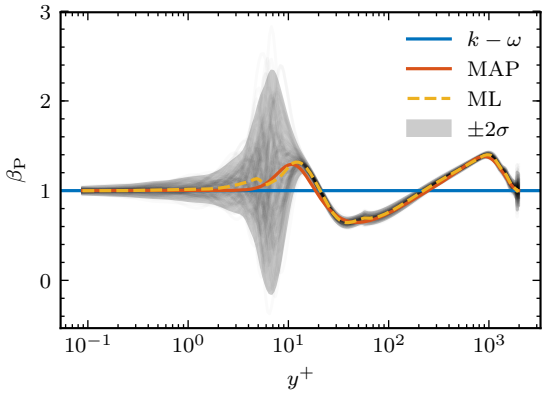
Table 8.1: Machine learning experiments.

8.1. Reynolds number generalization

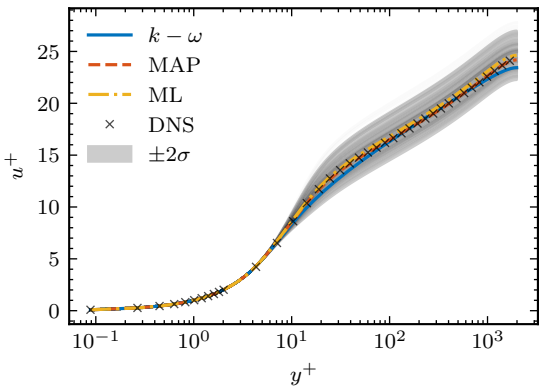
8.1.1. Turbulent channel flow

In the first experiment, both the training and the prediction is done for the turbulent channel flow case. The Gaussian process is trained on all available Reynolds numbers except for Re = 2,000, which is excluded from the training set. After pre-processing, the selected features are [F1.1, F1.3, F1.5, F2.1, F2.3, F3.2, F3.3, F3.5, F3.6, F3.8, F3.9]. The similarity between the datapoints is quantified using a radial basis function kernel. Optimizing the log-marginal likelihood resulted in a lengthscale of $l = 1.89$. The sampled corrective term predictions are compared to the MAP solution in fig. 8.1a, including $\pm 2\sigma$ uncertainty bounds. It can be noted that the prediction agrees well with the MAP solution in most regions. The main discrepancy is located around $y^+ = 1$, and is well reflected in the uncertainty bounds.

The mean velocity profiles resulting from propagating the sampled corrective term predictions is shown in fig. 8.1b, where they are compared to the $k-\omega$ model, the MAP solution, and the DNS data. The machine learning result provides an improvement over the base model for almost all distances from the wall. The discrepancy with respect to the data is slightly higher compared to that of the MAP solution and the data. The main difference is observed near the center of the channel, where it slightly overpredicts the streamwise velocity. Again, the predicted uncertainty is consistent with the discrepancy between the velocity profile resulting from the corrective term predicted by the Gaussian process and that of the MAP solution. Close to the wall, the corrective term is not effective and the baseline model is accurate. This is well reflected by the low uncertainty in this region. Near the center of the channel, the high uncertainty is consistent with the relatively large discrepancy in mean velocity. It is only in the log layer that the uncertainty is excessively high, given the relatively low discrepancy with the data. For this case, Gaussian processes can thus be concluded to perform well in terms of predictive accuracy, uncertainty quantification, and computational complexity.

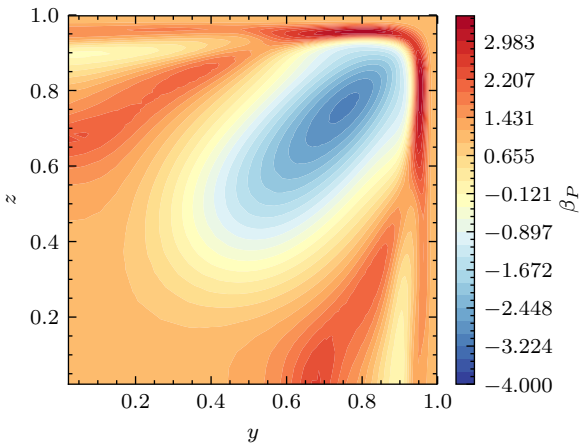


(a) Corrective term predicted by Gaussian process.

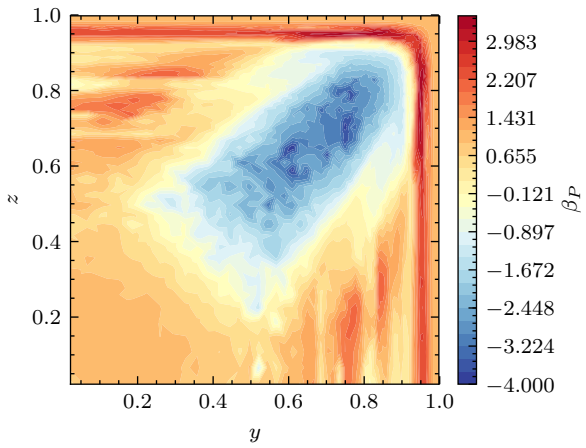


(b) Corrective function propagations.

Figure 8.1: $\mathrm{Re}_\tau = 2000$, model trained on $\mathrm{Re}_\tau \in \{180, 395, 550, 590, 950, 4200\}$.



(a) Inferred corrective term.



(b) Machine learning predicted corrective term.

Figure 8.2: Corrective term for square duct (Re = 3,500), random forest trained on square duct (Re = 1,100 – 2,900).

8.1.2. Square duct

In experiment II, the same approach as presented in the previous section is now applied to the square duct flow case. The machine learning algorithm is trained on Reynolds numbers varying from Re = 1,100 to Re = 2,900, and predicted on Re = 3,500. In this case, the higher number of features and the higher dimensionality of the dataset resulted in a larger dataset than in experiment I. Therefore, random forests were used from this experiment onward. The corrective term corresponding to the MAP solution is compared to the machine learning prediction in fig. 8.2. At least qualitatively, the two corrective terms are similar. It can be seen that the result predicted by the random forest is quite noisy compared to the inferred corrective term and the corrective term predicted by the Gaussian process for the turbulent channel flow. This might be improved by selecting a higher value for the minimum number of samples per leaf node. However, as the flow solver does not need to differentiate the production term, this does not result in problems in the numerical stability when propagating the machine learning prediction through the solver. It is expected that this is more difficult in the case where the machine learning is applied to the eigenvalue corrections, which are differentiated when the corrected anisotropy tensor is used in the momentum and k-equation. Figure 8.3 shows a quantitative comparison between the MAP solution for the corrective term and the machine learning prediction. The corrective terms match fairly well and there is no clear variation in discrepancy at specific values of the corrective term. From this is can be concluded that random forests can reproduce the corrective term for an unseen Reynolds number fairly accurately for this flow case.

The mean velocity profiles resulting from propagating the corrective term predicted by the machine learning algorithm is shown in fig. 8.4. It follows a similar trend to the MAP solution. Near the center of the channel,

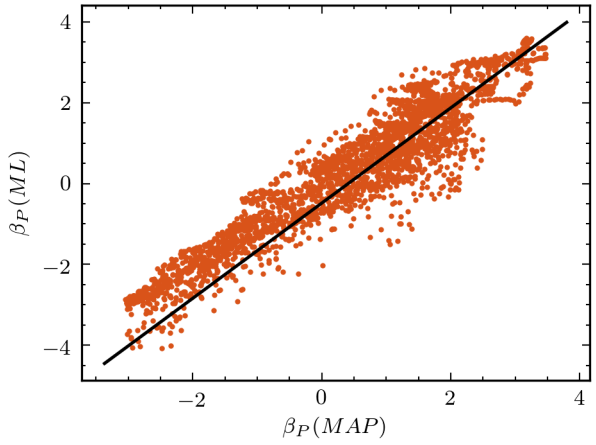


Figure 8.3: Comparison of inferred and predicted corrective term, square duct (Re = 3,500), random forest trained on square duct (Re = 1,100 – 2,900).

the velocity profile corresponding to the machine learning predicted corrective term does not provide a convincing improvement over the base model. However, closer to the walls, it is closer to the data than the base model. The higher axial velocity near the corner, which in the physical case originates from the increased momentum transfer due to the secondary motions, is still captured by the machine learning result. From this experiment it can be concluded that, even when the corrective terms have no direct physical interpretation, it may sometimes still be possible to use them in a machine learning setting within the same flow geometry. It could however be expected that the corrective functions obtained from this flow case do not generalize to other flows.

8.2. Geometry generalization

The ability of the corrective functions to be generalized across various flow geometries is investigated in experiments III and IV. The backward-facing step is used as test case, as it shows more severe separation than any of the cases in the training data. For this test case, the field inversion was not performed. However, the high-fidelity data can still be used to quantify the performance for the quantities of interest. Before presenting the results on the backward-facing step, it should first be noted that investigations using the square duct flow case in either the training or test set did not provide any improvements over the base model in any combination of test and training cases. This can again be explained by the fact that the corrective terms obtained from the square duct flow case do not have a physical meaning, and thus cannot be generalized beyond their usage for different Reynolds numbers on the same case.

The corrective term predicted by the random forest for experiment III in table 8.1 is shown in fig. 8.5. It can be seen that the corrective term is equal to one away from the shear layer. The predicted correction to the production term is largest in the shear layer, with a value below that corresponding to the base model. This is comparable to the inferred corrective terms of the cases in the training set, namely the periodic hills and the converging-diverging channel. One notable difference with respect to the inferred fields is that the predicted field is considerably less smooth. This again illustrates the advantage of applying the corrective term to the production term in the ω-equation. The fact that the corrective function is mainly decreased with respect to the value corresponding to the base model is consistent with the hypothesis that one effect of increasing the corrective term is to correct the separation location, as was the case for the converging-diverging channel. As the separation location is fixed in this case, the corrective term is decreased, similar to the inferred results for the periodic hills. The predicted corrective function is higher than one only in very specific locations. It might therefore be expected that more careful postprocessing of the predictions (e.g. applying a filter) might improve the results. The difference between the eddy viscosity resulting from propagating the corrective term predicted by the machine learning algorithm and that of the base model is shown in fig. 8.6. The noisy pattern of the corrective function is not reflected by the resulting eddy viscosity field. As expected, the main effect of the corrective function is to increase the eddy viscosity in regions where there is insufficient mixing. Again, this is in the region where the flow is separated, as the size of this region is overpredicted by the $k-\omega$ model.

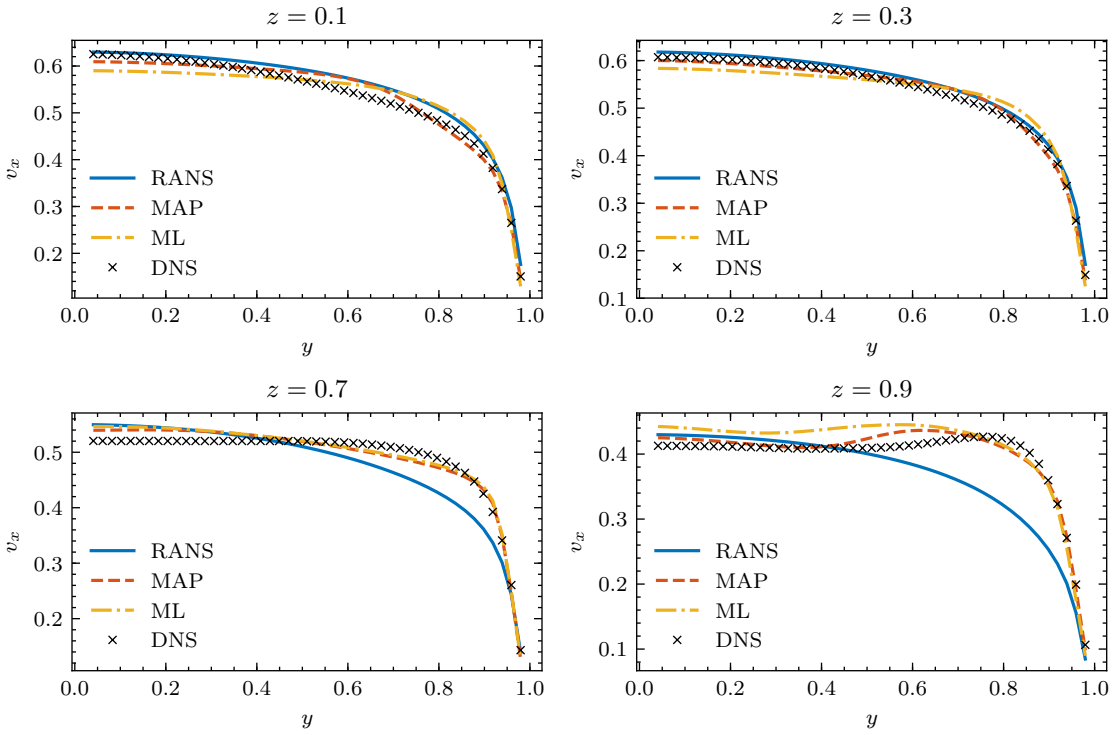


Figure 8.4: Velocity profiles resulting from propagation of predicted corrective term, square duct (Re = 3,500).

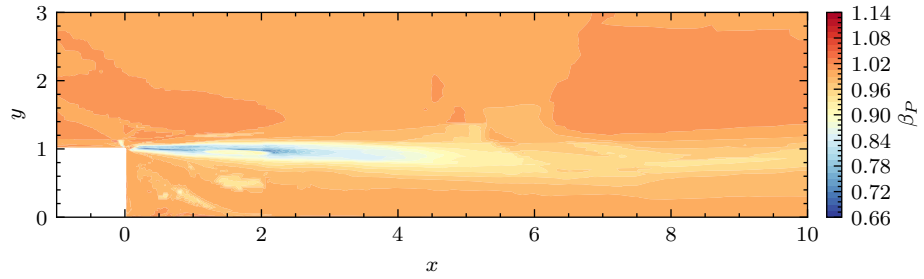


Figure 8.5: Predicted corrective function, backward-facing step (Re = 5,100).

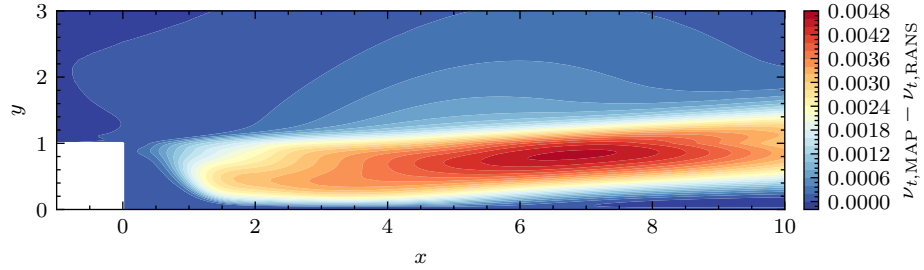


Figure 8.6: Difference in eddy viscosity between machine learning result and $k-\omega$ result, backward-facing step (Re = 5,100).

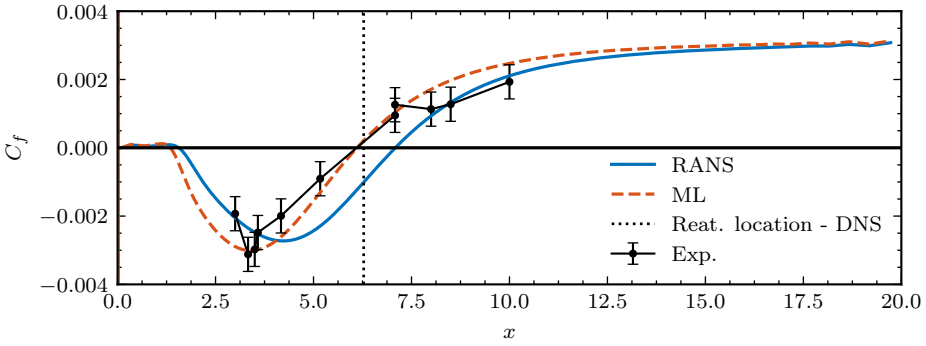


Figure 8.7: Skin friction coefficient, backward facing step, experimental data from Jovic and Driver [66] (Re = 5,100).

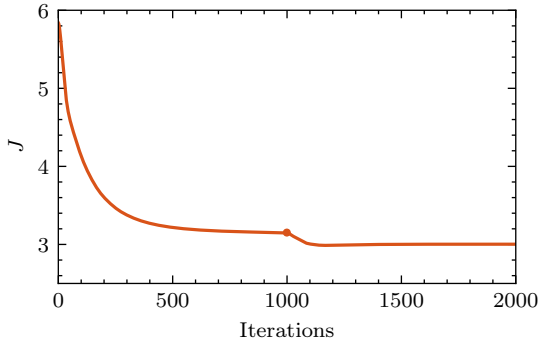


Figure 8.8: Effect of a second machine learning correction in experiment III, the marker indicates the iteration at which the second correction is introduced.

Finally, the skin friction coefficient resulting from propagating the machine learning prediction is shown in fig. 8.7. The result is compared with the result from the $k-\omega$ model and experimental data by Jovic and Driver [66]. The main quantity of interest, the reattachment location, is located at $x = 6.28$ for the DNS study by Le et al. [77]. The value found in the experiment is $x = 6 \pm 0.15$. The propagated machine learning result provides a clear improvement in the reattachment location with respect the base model. The skin-friction coefficient is especially improved near the reattachment location.

An interesting observation can be made when comparing experiment III with experiment IV, where the prediction is done for the same flow case but the periodic hill flow cases have been left out of the dataset. It could be hypothesised that including the periodic hill flow cases provides information about the relation between the machine learning features and the corrective term in regions where the flow is separated. This information is not expected to be present, or to a lesser extent, in the converging-diverging channel flow case, where only a small region of separation exists. It can indeed be seen that training the algorithm on only the converging-diverging channel results in only a small improvement when the predicted corrective term is propagated in the backward-facing step flow case, which features a large separated region.

8.3. Effect of performing multiple machine learning corrections

In all experiments described above, the corrective term predicted by the machine learning algorithm was only propagated through the flow solver once. As discussed in section 6.5, the iterative approach poses considerable challenges to the stability of the propagation solver. However, in some cases it was found that multiple corrections could give slightly better results than a single correction. In these cases, the new correction was only introduced after the equations including the preceding correction had been sufficiently converged. The convergence of the objective function is shown for experiment III in fig. 8.8, where the second correction is applied at the indicated iteration. It can be seen that the second correction yields a slight improvement. A third correction did not result in a further decrease in the objective function.

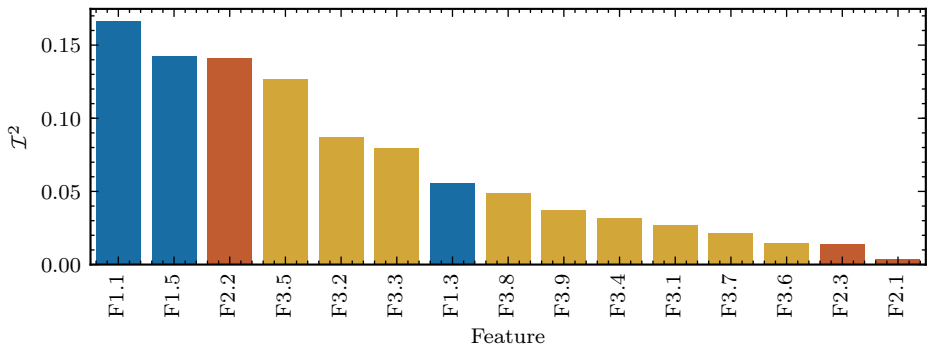


Figure 8.9: Squared relative importances for random forest trained on periodic hills (Re = 5,600 and 10,595) and converging-diverging channel (Re = 12,600).

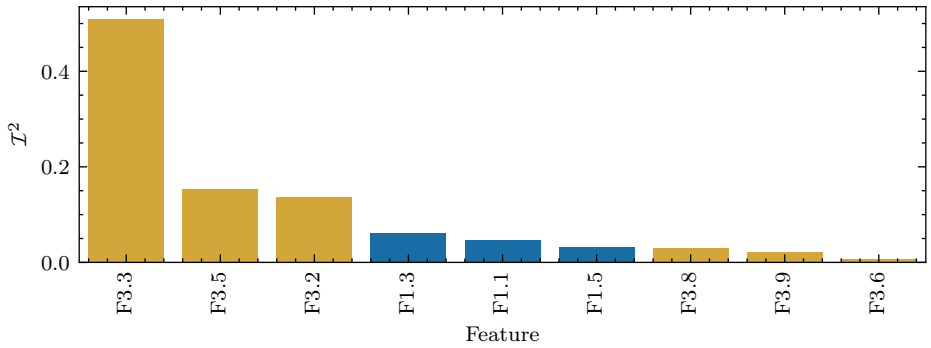


Figure 8.10: Squared relative importances for random forest trained on square duct (Re = 1,100 – 2,900).

8.4. Feature importance

As discussed in section 6.5, random forests allow the computation of the relative feature importance of each feature. The squared relative importances for the random forest trained on the periodic hills at both Reynolds numbers, the converging-diverging channel, and the turbulent channel flow at all Reynolds numbers (experiment III) are shown in fig. 8.9. The expression corresponding to each feature name is given in table 6.2. The two most important features both come from feature set 1. Furthermore, all features from feature set 3 were selected and have a relatively high importance. This illustrates the importance of including features constructed from physical reasoning. It can also be noted that a relatively low number of features from feature set 2 are selected. The relative feature importance for the random forest trained on the square duct flow case at various Reynolds numbers (experiment II) is shown in fig. 8.10. The number of selected features is considerably smaller than those for experiment IV. One reason for this is that the pressure is ill-defined for this flow case. Therefore, the features based on the pressure gradient (F3.4 and F3.6) do not provide useful information. Furthermore, many features in feature set 1 and feature set 2 are zero or each others negatives because of the zero velocity derivatives of the square duct flow case. It can also be noted that, in contrast to the other case, there is one main feature while the other features have a relatively low importance. This feature (F3.3) is the wall-distance based Reynolds number.

9

Conclusions and recommendations

9.1. Conclusions

Q1 Can the paradigm of field inversion and machine learning in its current form, as proposed by Parish and Duraisamy [107] and Singh et al. [138], be extended using the continuous adjoint of the $k-\omega$ model for the field inversion phase?

The continuous adjoint of the $k-\omega$ model was derived and implemented in OpenFOAM. The similarity between the primal and adjoint $k-\omega$ model allowed for a convenient implementation based on the implementation of the primal model. The adjoint model showed good convergence properties. Comparing the one-shot and complete approach for the optimization phase, it was found that the one-shot approach was both faster and more robust for the cases considered. For the machine learning phase, it was found that the iterative approach did not show convergent behavior in cases where the corrective approach yielded a significant improvement in the mean velocities and related quantities of interest.

The paradigm was evaluated on the possibility of quantifying the uncertainty in the corrective term and the resulting quantities of interest. For the production term correction in the turbulent channel flow case, it was found that a high uncertainty was predicted in regions where the production term was not effective and where thus no reliable quantity could be inferred. However, multiple assumptions and approximations are made of which the impact on the uncertainty bounds is unclear for the more complex cases. Furthermore, there are still challenges in making the uncertainty quantification computationally tractable. In the field inversion phase, even with a Gauss-Newton approximation for the Hessian, obtaining the posterior covariance is relatively expensive. In the machine learning phase, Gaussian processes were used for their ability to quantify the uncertainty in the predictions. The resulting uncertainty bounds were consistent with the discrepancy between the velocity resulting from propagating the machine learning prediction and the data. However, this algorithm does not scale well to larger flow cases.

Q2 What are the strengths and limitations of formulating the corrective term as corrections to the production term in the ω-equation and the eigenvalues of the Reynolds stress tensor?

For most cases, it can be concluded that the optimization process resulted in accurately inferred velocities and related quantities of interest. No significant differences were observed between the two considered formulations in terms of the accuracy. It was only in the cases where the discrepancies with the data were inherent to the forward model formulation that the more general corrections to the eigenvalues yielded a significant improvement over the production term correction. This was demonstrated in the square duct flow case, where the secondary flows are caused by the differences in normal stress, which cannot be modeled using LEVMs. However, even with ad-hoc modifications to the objective functions, the inferred in-plane velocities were still rather inaccurate.

A second criterion on which the corrective terms were evaluated was their physical interpretability. In cases where the corrective term did not improve upon the deficiencies in the base model of modeling the underlying physics, the resulting corrective function did not provide any interpretable information, as the corrective term is optimized to compensate for an effect which is not covered by the correction term. In other cases, the production term caused the augmented model to reduce to the base model in regions where it was sufficiently accurate. For the corrections to the eigenvalues, the resulting componentiality of the turbulence

could be compared with that of the data. It can be concluded that in all cases, there was a large discrepancy between the inferred eigenvalues and those of the stress tensor from the data. This can be explained by the non-unique relation between the eigenvalues of the stress tensor and the resulting mean velocities. It was hypothesised that this might be improved upon by considering a more accurate description of the observational covariance.

Thirdly, it was investigated whether the corrective terms would generalize in a machine learning setting. For the turbulent channel flow and square duct, a machine learning algorithm was trained on a variety of Reynolds numbers. It was observed that the production term correction could be predicted on an unseen higher Reynolds number fairly accurately. Propagating the corrective term through the flow solver yielded improvements in terms of the mean velocities. Generalizing the corrective term to other flow geometries proved to be more challenging. The corrective terms can therefore be concluded not to be learnable in all cases. However, in several flow cases the propagated machine learning prediction yielded an improvement in the mean velocities. Also, it was found that including in the training set a case with similar flow features as the test case improved the results significantly.

9.2. Recommendations for future work

Field inversion In the field inversion phase, there are several choices which can be further investigated in future work. In this work, only the discrepancy of the mean velocities was considered in the objective function. It would be interesting to see how different formulations of the objective function influence the results of the field inversion. Apart from the quantities of interest included, the choice of the observational covariance matrix and the prior also affect the objective function. In the preliminary investigations in the model problem, it was seen that an accurately formulated observational covariance matrix had a considerable influence on the physical interpretability of the corrective term and the accuracy of the posterior covariance. Investigations into methods to construct a more accurate observational covariance matrix from DNS or experimental data might therefore be interesting to include in future research. Inspiration might be taken from the work of Wieneke and Sciacchitano [161] or [5]. Furthermore, more detailed specifications of the prior distributions might help ensure realizability of the result of the inversion process. For example, for quantities which can be bounded physically a distribution with finite support could be used to ensure realizability. One example would be to use a Beta or Dirichlet distribution as a prior for the eigenvalues or barycentric coordinates.

Another area of the paradigm which might benefit from further research is the uncertainty quantification. In the field inversion phase, this mainly concerns the approximation of the Hessian. Even for the simplest two-dimensional cases considered in this work, the approximation used in this work was excessively expensive. Furthermore, it is difficult to quantify the impact of the assumptions made and it is expected that the computational savings do not weigh up to the accuracy decrease. Nevertheless, it would be interesting to study the impact of the Gauss-Newton assumption on more complex flow cases.

The options for the formulations of the corrective term are practically endless. One example of a suggestion for future research would be to also infer the correction to the eigenvectors. If the magnitude and the shape would also be corrected, this would allow to infer corrections to the complete Reynolds stress tensor. A possible formulation which includes the correction to the eigenvectors and ensures realizability of the corrected Reynolds stress tensor is introduced in appendix B.6.

Machine learning The problems with scaling the uncertainty quantification which occurred in the field inversion also exist in the machine learning phase. Gaussian processes are convenient because of their inherently probabilistic nature. However, this algorithm is not computationally feasible in larger flow cases. One way to alleviate this issue would be the use of hierarchical Gaussian processes [121, 170, 171]. Furthermore, the use of other stochastic machine learning algorithms could be investigated. An example could be Bayesian neural networks [12].

Apart from the probabilistic capabilities of the machine learning algorithms, it could be also be interesting to consider the recent developments in machine learning algorithms for data-driven turbulent modeling taking into account invariance in the machine learning phase. For example, tensor basis neural networks proposed by Ling et al. [82] or tensor basis random forests described by Kaandorp and Dwight [68] could be used. Furthermore, the field of data-driven turbulence modeling might benefit from further research in the iterative approach. This would allow moving from data-driven models being just one corrective step to a more integrated approach between the base model and the data-driven enhancement.

A natural next step for the complete paradigm would be to test it on a more complex or more diverse set

of training and test cases. This could result in complications in several parts of the paradigm. For example, whereas the one-shot approach in the optimization phase had a robust and convergent performance for the cases considered in this work, more robust alternative may be needed for more complex cases. Modifications to the one-shot approach could be the use of gradient smoothing, which has been shown by Jameson and Vassberg [61] to improve the convergence of the optimization. Another option would be to further investigate the complete approach coupled with a more robust optimization algorithm. For the machine learning phase, the improved mean fields might benefit from more elaborate post-processing of the predicted corrective functions. For the random forests, it was observed that the predicted corrective terms were relatively noisy. Especially if corrections to the anisotropy tensor are used in the machine learning phase, this might pose difficulties when propagating the predictions in the solver.

A

Discrete direct-adjoint Hessian

This appendix gives the derivation of the Hessian using the discrete direct-adjoint approach, as it is used in the model problem in chapter 3. The derivation and notation is based on the work by Papadimitriou and Giannakoglou [104]. The general derivation given in appendix A.1 is applied to the model problem in appendix A.2.

A.1. General derivation

The derivative of the objective function and the governing equations can be rewritten using the chain rule:

$$\frac{\mathrm{d}J}{\mathrm{d}\beta_i} = \frac{\partial J}{\partial \beta_i} + \frac{\partial J}{\partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i} \tag{A.1}$$

and

$$\frac{\mathrm{d}R_m}{\mathrm{d}\beta_i} = \frac{\partial R_m}{\partial \beta_i} + \frac{\partial R_m}{\partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i} = 0. \tag{A.2}$$

Similarly, the second derivative can be rewritten as

$$\frac{\mathrm{d}^2 J}{\mathrm{d}\beta_i \mathrm{d}\beta_j} = \frac{\partial^2 J}{\partial \beta_i \partial \beta_j} + \frac{\partial^2 J}{\partial \beta_i \partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_j} + \frac{\partial^2 J}{\partial \beta_j \partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i} + \frac{\partial^2 J}{\partial T_k \partial T_m}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i}\frac{\mathrm{d}T_m}{\mathrm{d}\beta_j} + \frac{\partial J}{\partial T_k}\frac{\mathrm{d}^2 T_k}{\mathrm{d}\beta_i \mathrm{d}\beta_j} \tag{A.3}$$

and

$$\frac{\mathrm{d}^2 R_n}{\mathrm{d}\beta_i \mathrm{d}\beta_j} = \frac{\partial^2 R_n}{\partial \beta_i \partial \beta_j} + \frac{\partial^2 R_n}{\partial \beta_i \partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_j} + \frac{\partial^2 R_n}{\partial \beta_j \partial T_k}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i} + \frac{\partial^2 R_n}{\partial T_k \partial T_m}\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i}\frac{\mathrm{d}T_m}{\mathrm{d}\beta_j} + \frac{\partial R_n}{\partial T_k}\frac{\mathrm{d}^2 T_k}{\mathrm{d}\beta_i \mathrm{d}\beta_j} = 0. \tag{A.4}$$

Introducing a new augmented objective function $\hat{J}$ and a new vector of adjoint variables $\hat{\Psi}$,

$$\frac{\mathrm{d}^2 \hat{J}}{\mathrm{d}\beta_i \mathrm{d}\beta_j} = \frac{\partial^2 J}{\partial \beta_i \partial \beta_j} + \hat{\Psi}_n \frac{\mathrm{d}^2 R_n}{\mathrm{d}\beta_i \mathrm{d}\beta_j}. \tag{A.5}$$

Substituting the derivatives on the right hand side from (A.3) and (A.4) and rearranging the terms gives

$$\begin{aligned}\frac{\mathrm{d}^2 \hat{J}}{\mathrm{d}\beta_i \mathrm{d}\beta_j} = {} & \frac{\partial^2 J}{\partial \beta_i \partial \beta_j} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial \beta_i \partial \beta_j} + \left(\frac{\partial^2 J}{\partial T_k \partial T_m} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial T_k \partial T_m}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i}\frac{\mathrm{d}T_m}{\mathrm{d}\beta_j} \\ & + \left(\frac{\partial^2 J}{\partial \beta_i \partial T_k} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial \beta_i \partial T_k}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_j} + \left(\frac{\partial^2 J}{\partial \beta_j \partial T_k} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial T_k \partial \beta_j}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i} \\ & + \left(\frac{\partial J}{\partial T_k} + \hat{\Psi}_n \frac{\partial R_n}{\partial T_k}\right)\frac{\mathrm{d}^2 T_k}{\mathrm{d}\beta_i \mathrm{d}\beta_j}.\end{aligned} \tag{A.6}$$

By satisfying the adjoint equation

$$\frac{\partial J}{\partial T_k} + \hat{\Psi}_n \frac{\partial R_n}{\partial T_k} = 0, \tag{A.7}$$

the Hessian is given by

$$\begin{aligned}\frac{\mathrm{d}^2\hat{J}}{\mathrm{d}\beta_i\mathrm{d}\beta_j} = \frac{\partial^2 J}{\partial\beta_i\partial\beta_j} + \hat{\Psi}_n\frac{\partial^2 R_n}{\partial\beta_i\partial\beta_j} + \left(\frac{\partial^2 J}{\partial T_k\partial T_m} + \hat{\Psi}_n\frac{\partial^2 R_n}{\partial T_k\partial T_m}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i}\frac{\mathrm{d}T_m}{\mathrm{d}\beta_j} \\ + \left(\frac{\partial^2 J}{\partial\beta_i\partial T_k} + \hat{\Psi}_n\frac{\partial^2 R_n}{\partial\beta_i\partial T_k}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_j} + \left(\frac{\partial^2 J}{\partial\beta_j\partial T_k} + \hat{\Psi}_n\frac{\partial^2 R_n}{\partial T_k\partial\beta_j}\right)\frac{\mathrm{d}T_k}{\mathrm{d}\beta_i},\end{aligned} \tag{A.8}$$

where $\mathrm{d}T_k/\mathrm{d}\beta_i$ is obtained by solving (A.2)

A.2. Model problem

For the one-dimensional model problem introduced in chapter 3, the second explicit derivatives can be derived straightforwardly from (3.42)–(3.45). The resulting expressions are:

$$\frac{\partial^2 J}{\partial\beta_i\partial\beta_k} = (C_\beta^{-1})_{ij}\delta_{jk} \tag{A.9}$$

$$\frac{\partial^2 J}{\partial\beta_i\partial T_j} = 0 \tag{A.10}$$

$$\frac{\partial^2 J}{\partial T_i\partial T_m} = H_{km}(C_m^{-1})_{jk}H_{ji} \tag{A.11}$$

$$\frac{\partial^2 R_\alpha}{\partial\beta_i\partial\beta_j} = 0 \tag{A.12}$$

$$\frac{\partial^2 R_\alpha}{\partial\beta_i\partial T_j} = 2(\Delta z)^2\delta_{\alpha i}\varepsilon_0 T_\alpha^3\delta_{\alpha j} \tag{A.13}$$

$$\frac{\partial^2 R_\alpha}{\partial T_i\partial T_j} = \delta_{\alpha i}\delta_{\alpha j}6T_\alpha^2(\Delta z)^2\beta_\alpha\varepsilon_0. \tag{A.14}$$

B

Continuous adjoint of the $k-\omega$ model

B.1. Adjoint derivation

First, the adjoint equations will be derived for the governing equations without any corrective terms. After the adjoint equations have been obtained, the additions due to the corrective terms will be derived separately. The governing equations are given by

$$R_i^v = \frac{\partial(v_i v_j)}{\partial x_j} + \frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu+\nu_t)\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\right) = 0, \tag{B.1}$$

$$R^p = -\frac{\partial v_j}{\partial x_j} = 0, \tag{B.2}$$

$$R^k = \frac{\partial(v_j k)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial k}{\partial x_j}\right) - P_k + \frac{2}{3}\frac{\partial v_j}{\partial x_j}k + C_\mu \omega k = 0, \tag{B.3}$$

$$R^\omega = \frac{\partial(v_j \omega)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega \nu_t + \nu)\frac{\partial \omega}{\partial x_j}\right) - \gamma P + \frac{2}{3}\gamma\frac{\partial v_j}{\partial x_j}\omega + \alpha\omega^2 = 0, \tag{B.4}$$

where

$$P_k = \nu_t\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}, \tag{B.5}$$

and

$$P = \nu_t\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right). \tag{B.6}$$

The objective function J is decomposed into a domain and a boundary integral.

$$J = \int_\Omega J_\Omega \,\mathrm{d}\Omega + \int_\Gamma J_\Gamma \,\mathrm{d}\Gamma \tag{B.7}$$

In contrast to applications of adjoint theory in shape optimization, in this application the geometry is independent of β. The sensitivity of the augmented objective function can then be written as

$$\frac{\delta L}{\delta \beta} = \int_\Omega \frac{\delta J_\Omega}{\delta \beta}\,\mathrm{d}\Omega + \int_\Gamma \frac{\delta J_\Gamma}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega u_i \frac{\delta R_i^v}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega q\frac{\delta R^p}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega k_a \frac{\delta R^k}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega \omega_a \frac{\delta R^\omega}{\delta \beta}\,\mathrm{d}\Omega\,. \tag{B.8}$$

The sensitivity of the objective function J can be expanded as

$$\int_\Omega \frac{\delta J_\Omega}{\delta \beta}\,\mathrm{d}\Omega = \int_\Omega \frac{\partial J_\Omega}{\partial \beta} + \frac{\partial J_\Omega}{\partial v_i}\frac{\delta v_i}{\delta \beta} + \frac{\partial J_\Omega}{\partial p}\frac{\delta p}{\delta \beta} + \frac{\partial J_\Omega}{\partial k}\frac{\delta k}{\delta \beta} + \frac{\partial J_\Omega}{\partial \omega}\frac{\delta \omega}{\delta \beta}, \tag{B.9}$$

$$\int_\Gamma \frac{\delta J_\Gamma}{\delta \beta}\,\mathrm{d}\Gamma = \int_\Gamma \frac{\partial J_\Gamma}{\partial \beta} + \frac{\partial J_\Gamma}{\partial v_i}\frac{\delta v_i}{\delta \beta} + \frac{\partial J_\Gamma}{\partial p}\frac{\delta p}{\delta \beta} + \frac{\partial J_\Gamma}{\partial k}\frac{\delta k}{\delta \beta} + \frac{\partial J_\Gamma}{\partial \omega}\frac{\delta \omega}{\delta \beta}. \tag{B.10}$$

The sensitivity of the production terms and the eddy viscosity can be derived as

$$\frac{\delta P_k}{\delta \beta} = \frac{\delta \nu_t}{\delta \beta}\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j} + 2\nu_t\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta \beta}\right), \tag{B.11}$$

$$\frac{\delta P}{\delta\beta} = 2\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right), \tag{B.12}$$

and

$$\frac{\delta \nu_t}{\delta\beta} = \frac{\delta}{\delta\beta}\left(\frac{k}{\omega}\right) = \frac{1}{\omega}\frac{\delta k}{\delta\beta} - \frac{k}{\omega^2}\frac{\delta\omega}{\delta\beta}. \tag{B.13}$$

The term in (B.8) involving the momentum equation can now be derived by considering the sensitivity of the momentum equation.

$$\begin{aligned}\frac{\delta R_i^v}{\delta\beta} = {}& \frac{\partial v_i}{\partial x_j}\frac{\delta v_j}{\delta\beta} + v_j\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) + \frac{\partial v_j}{\partial x_j}\frac{\delta v_i}{\delta\beta} + v_i\frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta\beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta p}{\delta\beta}\right) \\ & - \frac{\partial}{\partial x_j}\left(\frac{\delta \nu_t}{\delta\beta}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) - \frac{\partial}{\partial x_j}\left((\nu+\nu_t)\left(\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\right)\right) = 0\end{aligned} \tag{B.14}$$

Multiplying by the adjoint velocity and integrating over the domain gives

$$\begin{aligned}\int_\Omega u_i\frac{\delta R_i^v}{\delta\beta}\,\mathrm{d}\Omega = \int_\Omega u_i\frac{\partial v_i}{\partial x_j}\frac{\delta v_j}{\delta\beta}\underbrace{+u_i v_j\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)}_{\mathrm{I}} + u_i\frac{\partial v_j}{\partial x_j}\frac{\delta v_i}{\delta\beta}\underbrace{+u_i v_i\frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta\beta}\right)}_{\mathrm{II}}\underbrace{+u_i\frac{\partial}{\partial x_i}\left(\frac{\delta p}{\delta\beta}\right)}_{\mathrm{III}} \\ \underbrace{-u_i\frac{\partial}{\partial x_j}\left\{(\nu+\nu_t)\left[\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\right]\right\}}_{\mathrm{IV}}\underbrace{-u_i\frac{\partial}{\partial x_j}\left[\frac{\delta \nu_t}{\delta\beta}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]}_{\mathrm{V}}\mathrm{d}\Omega\,.\end{aligned} \tag{B.15}$$

The terms which need to be expanded using integration by parts are indicated, and are derived below.

$$\mathrm{I} = \int_\Gamma u_i v_j n_j\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial(u_i v_j)}{\partial x_j}\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Omega \tag{B.16}$$

$$\mathrm{II} = \int_\Gamma u_i v_i n_j\frac{\delta v_j}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial(u_i v_i)}{\partial x_j}\frac{\delta v_j}{\delta\beta}\,\mathrm{d}\Omega \tag{B.17}$$

$$\mathrm{III} = \int_\Gamma u_i n_i\frac{\delta p}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial u_i}{\partial x_i}\frac{\delta p}{\delta\beta}\,\mathrm{d}\Omega \tag{B.18}$$

$$\begin{aligned}\mathrm{IV.A} &= -\int_\Omega u_i\frac{\partial}{\partial x_j}\left((\nu+\nu_t)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\right)\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j(\nu+\nu_t)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\mathrm{d}\Gamma + \int_\Omega\frac{\partial u_i}{\partial x_j}(\nu+\nu_t)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j(\nu+\nu_t)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\mathrm{d}\Gamma + \int_\Gamma\frac{\partial u_i}{\partial x_j}(\nu+\nu_t)n_j\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial}{\partial x_j}\left((\nu+\nu_t)\frac{\partial u_i}{\partial x_j}\right)\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.19}$$

$$\begin{aligned}\mathrm{IV.B} &= -\int_\Omega u_i\frac{\partial}{\partial x_j}\left((\nu+\nu_t)\frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\right)\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j(\nu+\nu_t)\frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\mathrm{d}\Gamma + \int_\Omega\frac{\partial u_i}{\partial x_j}(\nu+\nu_t)\frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j(\nu+\nu_t)\frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\mathrm{d}\Gamma + \int_\Gamma\frac{\partial u_i}{\partial x_j}(\nu+\nu_t)n_i\frac{\delta v_j}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial}{\partial x_i}\left((\nu+\nu_t)\frac{\partial u_i}{\partial x_j}\right)\frac{\delta v_j}{\delta\beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.20}$$

$$\begin{aligned}\mathrm{V} &= -\int_\Omega u_i\frac{\partial}{\partial x_j}\left[\frac{\delta \nu_t}{\delta\beta}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\delta \nu_t}{\delta\beta}\,\mathrm{d}\Gamma + \int_\Omega\frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\delta \nu_t}{\delta\beta}\,\mathrm{d}\Omega \\ &= -\int_\Gamma u_i n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{1}{\omega}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Gamma + \int_\Gamma u_i n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{k}{\omega^2}\frac{\delta\omega}{\delta\beta}\,\mathrm{d}\Gamma \\ &\quad + \int_\Omega\frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{1}{\omega}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Omega - \int_\Omega\frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{k}{\omega^2}\frac{\delta\omega}{\delta\beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.21}$$

A similar procedure is used for the remaining governing equations. Starting with the continuity equation, the derivative can be written as

$$\frac{\delta R^p}{\delta \beta} = -\frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right). \tag{B.22}$$

Multiplying by the adjoint pressure, integrating over the domain, and applying integration by parts gives

$$\int_\Omega q \frac{\delta R^p}{\delta \beta}\,\mathrm{d}\Omega = -\int_\Omega q \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right)\mathrm{d}\Omega = -\int_\Gamma q n_i \frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial q}{\partial x_i}\frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Omega. \tag{B.23}$$

The sensitivity of the ω equation with respect to the corrective function can be written as

$$\begin{aligned}\frac{\delta R^\omega}{\delta \beta} = {} & \frac{\partial \omega}{\partial x_j}\frac{\delta v_j}{\delta \beta} + v_j \frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right) + \frac{\partial v_j}{\partial x_j}\frac{\delta \omega}{\delta \beta} + \omega \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right) - \frac{\partial}{\partial x_j}\left(\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{\delta v_t}{\delta \beta}\right) \\ & - \frac{\partial}{\partial x_j}\left((\sigma_\omega v_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\right) - \gamma \frac{\delta P}{\delta \beta} + \frac{2}{3}\gamma\omega \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right) + \frac{2}{3}\gamma \frac{\partial v_j}{\partial x_j}\frac{\delta \omega}{\delta \beta} + 2\alpha\omega\frac{\delta \omega}{\delta \beta}.\end{aligned} \tag{B.24}$$

The resulting expression is multiplied by the adjoint specific turbulent dissipation rate ω_a and integrated over the domain.

$$\begin{aligned}\int_\Omega \omega_a \frac{\delta R^\omega}{\delta \beta}\,\mathrm{d}\Omega = {} & \int_\Omega \omega_a \frac{\partial \omega}{\partial x_j}\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \underbrace{+ \int_\Omega \omega_a v_j \frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{I}} + \int_\Omega \omega_a \frac{\partial v_j}{\partial x_j}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega \underbrace{+ \int_\Omega \omega_a \omega \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right)}_{\mathrm{II}} \\ & \underbrace{- \int_\Omega \omega_a \frac{\partial}{\partial x_j}\left(\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{\delta v_t}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{III}} \underbrace{- \int_\Omega \omega_a \frac{\partial}{\partial x_j}\left((\sigma_\omega v_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\right)\mathrm{d}\Omega}_{\mathrm{IV}} \\ & \underbrace{- \int_\Omega \omega_a \gamma \frac{\delta P}{\delta \beta}\,\mathrm{d}\Omega}_{\mathrm{V}} \underbrace{+ \int_\Omega \frac{2}{3}\omega_a \gamma \omega \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{VI}} + \int_\Omega \frac{2}{3}\omega_a \gamma \frac{\partial v_j}{\partial x_j}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega \\ & + \int_\Omega 2\omega_a \alpha \omega \frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.25}$$

The indicated terms are now rewritten using integration by parts.

$$\mathrm{I} = \int_\Gamma \omega_a v_j n_j \frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial(\omega_a v_j)}{\partial x_j}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega \tag{B.26}$$

$$\mathrm{II} = \int_\Gamma \omega_a \omega n_j \frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial(\omega_a \omega)}{\partial x_j}\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \tag{B.27}$$

$$\begin{aligned}\mathrm{III} = {} & -\int_\Gamma \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{\delta v_t}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial \omega_a}{\partial x_j}\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{\delta v_t}{\delta \beta}\,\mathrm{d}\Omega \\ = {} & -\int_\Gamma \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{1}{\omega}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Gamma \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{k}{\omega^2}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial \omega_a}{\partial x_j}\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{1}{\omega}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega - \int_\Omega \frac{\partial \omega_a}{\partial x_j}\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{k}{\omega^2}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.28}$$

$$\begin{aligned}\mathrm{IV} = {} & -\int_\Gamma \omega_a n_j (\sigma_\omega v_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\mathrm{d}\Gamma + \int_\Omega \frac{\partial \omega_a}{\partial x_j}(\sigma_\omega v_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\mathrm{d}\Omega \\ = {} & -\int_\Gamma \omega_a n_j (\sigma_\omega v_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\mathrm{d}\Gamma + \int_\Gamma \frac{\partial \omega_a}{\partial x_j}(\sigma_\omega v_t + \nu) n_j \frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial}{\partial x_j}\left((\sigma_\omega v_t + \nu)\frac{\partial \omega_a}{\partial x_j}\right)\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.29}$$

$$\begin{aligned}\mathrm{V} = {} & -\int_\Omega 2\omega_a \gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta \beta}\right)\mathrm{d}\Omega \\ = {} & -\int_\Gamma 2\omega_a \gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) n_j \frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial}{\partial x_j}\left(2\omega_a \gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right)\frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.30}$$

$$\mathrm{VI} = \int_\Gamma \frac{2}{3}\omega_a \gamma \omega n_j \frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial}{\partial x_j}\left(\frac{2}{3}\omega_a \gamma \omega\right)\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \tag{B.31}$$

Finally, the sensitivity of the k-equation can be written as

$$\begin{aligned}\frac{\delta R^k}{\delta \beta} = {} & \frac{\partial k}{\partial x_j}\frac{\delta v_j}{\delta \beta} + v_j \frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right) + \frac{\partial v_j}{\partial x_j}\frac{\delta k}{\delta \beta} + k\frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right) - \frac{\partial}{\partial x_j}\left(\sigma_k \frac{\partial k}{\partial x_j}\frac{\delta \nu_t}{\delta \beta}\right) \\ & - \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\right) - \frac{\delta P_k}{\delta \beta} + \frac{2}{3}k\frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right) + \frac{2}{3}\frac{\partial v_j}{\partial x_j}\frac{\delta k}{\delta \beta} + C_\mu k\frac{\delta \omega}{\delta \beta} + C_\mu \omega \frac{\delta k}{\delta \beta}\end{aligned} \tag{B.32}$$

Again, the expression is multiplied by the adjoint turbulent kinetic energy k_a and integrated over the domain.

$$\begin{aligned}\int_\Omega k_a \frac{\delta R^k}{\delta \beta}\,\mathrm{d}\Omega = {} & \int_\Omega k_a \frac{\partial k}{\partial x_j}\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \underbrace{+ \int_\Omega k_a v_j \frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{I}} + \int_\Omega k_a \frac{\partial v_j}{\partial x_j}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega \underbrace{+ \int_\Omega k_a k \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{II}} \\ & \underbrace{- \int_\Omega k_a \frac{\partial}{\partial x_j}\left(\sigma_k \frac{\partial k}{\partial x_j}\frac{\delta \nu_t}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{III}} \underbrace{- \int_\Omega k_a \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\right)\mathrm{d}\Omega}_{\mathrm{IV}} \underbrace{- \int_\Omega k_a \frac{\delta P_k}{\delta \beta}\,\mathrm{d}\Omega}_{\mathrm{V}} \\ & \underbrace{+ \int_\Omega \frac{2}{3}k_a k \frac{\partial}{\partial x_j}\left(\frac{\delta v_j}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{VI}} + \int_\Omega \frac{2}{3}k_a \frac{\partial v_j}{\partial x_j}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega C_\mu k_a k \frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega C_\mu k_a \omega \frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.33}$$

In order to isolate the sensitivities of the primal variables, the indicated terms are rewritten using integration by parts.

$$\mathrm{I} = \int_\Gamma k_a v_j n_j \frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega - \int_\Omega \frac{\partial}{\partial x_j}\left(k_a v_j\right)\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega \tag{B.34}$$

$$\mathrm{II} = \int_\Gamma k_a k n_j \frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial}{\partial x_j}\left(k_a k\right)\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \tag{B.35}$$

$$\begin{aligned}\mathrm{III} = {} & -\int_\Gamma k_a n_j \sigma_k \frac{\partial k}{\partial x_j}\frac{\delta \nu_t}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial k_a}{\partial x_j}\sigma_k \frac{\partial k}{\partial x_j}\frac{\delta \nu_t}{\delta \beta}\,\mathrm{d}\Omega \\ = {} & -\int_\Gamma k_a n_j \sigma_k \frac{\partial k}{\partial x_j}\frac{1}{\omega}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Gamma k_a n_j \sigma_k \frac{\partial k}{\partial x_j}\frac{k}{\omega^2}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial k_a}{\partial x_j}\sigma_k \frac{\partial k}{\partial x_j}\frac{1}{\omega}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega - \int_\Omega \frac{\partial k_a}{\partial x_j}\sigma_k \frac{\partial k}{\partial x_j}\frac{k}{\omega^2}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.36}$$

$$\begin{aligned}\mathrm{IV} = {} & -\int_\Gamma k_a n_j (\sigma_k \nu_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\mathrm{d}\Gamma + \int_\Omega \frac{\partial k_a}{\partial x_j}(\sigma_k \nu_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\mathrm{d}\Omega \\ = {} & -\int_\Gamma k_a n_j (\sigma_k \nu_t + \nu)\frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right)\mathrm{d}\Gamma + \int_\Gamma \frac{\partial k_a}{\partial x_j}(\sigma_k \nu_t + \nu) n_j \frac{\delta k}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial k_a}{\partial x_j}\right)\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega\end{aligned} \tag{B.37}$$

$$\mathrm{V} = \underbrace{-\int_\Omega k_a \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{\delta \nu_t}{\delta \beta}\,\mathrm{d}\Omega}_{\mathrm{A}} \underbrace{- \int_\Omega 2k_a \nu_t \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta \beta}\right)\mathrm{d}\Omega}_{\mathrm{B}} \tag{B.38}$$

$$\mathrm{V.A} = -\int_\Omega k_a \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{1}{\omega}\frac{\delta k}{\delta \beta}\,\mathrm{d}\Omega + \int_\Omega k_a \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{k}{\omega^2}\frac{\delta \omega}{\delta \beta}\,\mathrm{d}\Omega \tag{B.39}$$

$$\mathrm{V.B} = -\int_\Gamma 2k_a \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) n_j \frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Gamma + \int_\Omega 2\frac{\partial}{\partial x_j}\left(k_a \nu_t \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right)\frac{\delta v_i}{\delta \beta}\,\mathrm{d}\Omega \tag{B.40}$$

$$\mathrm{VI} = \int_\Gamma \frac{2}{3} k_a k n_j \frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{2}{3}\frac{\partial (k_a k)}{\partial x_j}\frac{\delta v_j}{\delta \beta}\,\mathrm{d}\Omega \tag{B.41}$$

The sensitivities of the primal variables can now be isolated. This gives an expression of the following form.

$$\begin{aligned}\frac{\delta L}{\delta \beta} = &\int_\Omega \left(\frac{\partial J_\Omega}{\partial \beta} + R_i^u \frac{\delta v_i}{\delta \beta} + R^q \frac{\delta p}{\delta \beta} + R^{k_a}\frac{\delta k}{\delta \beta} + R^{\omega_a}\frac{\delta \omega}{\delta \beta}\right)\mathrm{d}\Omega \\ &+ \int_\Gamma \left(\frac{\partial J_\Gamma}{\partial \beta} + D_i^u \frac{\delta v_i}{\delta \beta} + D^q \frac{\delta p}{\delta \beta} + D^{k_a}\frac{\delta k}{\delta \beta} + D^{\omega_a}\frac{\delta \omega}{\delta \beta}\right. \\ &\left. + P_{ij}^u \frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta \beta}\right) + P_j^k \frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right) + P_j^\omega \frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right)\right)\mathrm{d}\Gamma\end{aligned} \tag{B.42}$$

The adjoint equations are obtained by setting the expressions multiplying the sensitivities of the primal variables in the domain integral to zero.

$$\begin{aligned}R_i^u = &\frac{\partial J_\Omega}{\partial v_i} - \frac{\partial u_i v_j}{\partial x_j} - v_j \frac{\partial u_j}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu + \nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) + \frac{\partial q}{\partial x_i} - k\frac{\partial k_a}{\partial x_i} - \omega \frac{\partial \omega_a}{\partial x_i} \\ &+ 2\frac{\partial}{\partial x_j}\left((\omega_a \gamma + k_a \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) - \frac{2}{3}\gamma \frac{\partial(\omega_a \omega)}{\partial x_i} - \frac{2}{3}\frac{\partial (k_a k)}{\partial x_i} = 0\end{aligned} \tag{B.43}$$

$$R^q = \frac{\partial J_\Omega}{\partial p} - \frac{\partial u_i}{\partial x_i} = 0 \tag{B.44}$$

$$\begin{aligned}R^{k_a} = &\frac{\partial J_\Omega}{\partial k} - \frac{\partial (k_a v_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_k \nu_t + \nu)\frac{\partial k_a}{\partial x_j}\right) + \frac{\partial k_a}{\partial x_j}\sigma_k \frac{\partial k}{\partial x_j}\frac{1}{\omega} + \frac{\partial \omega_a}{\partial x_j}\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{1}{\omega} + \frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{1}{\omega} \\ &- k_a\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{1}{\omega} + \frac{2}{3}k_a \frac{\partial v_j}{\partial x_j} - \frac{2}{3}\frac{\partial u_i}{\partial x_i} + C_\mu k_a \omega = 0\end{aligned} \tag{B.45}$$

$$\begin{aligned}R^{\omega_a} = &\frac{\partial J_\Omega}{\partial \omega} - \frac{\partial (\omega_a v_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega \nu_t + \nu)\frac{\partial \omega_a}{\partial x_j}\right) - \frac{\partial \omega_a}{\partial x_j}\sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{k}{\omega^2} + \frac{2}{3}\gamma \omega_a \frac{\partial v_j}{\partial x_j} + 2\alpha \omega_a \omega - \frac{\partial k_a}{\partial x_j}\sigma_k \frac{\partial k}{\partial x_j}\frac{k}{\omega^2} \\ &+ k_a\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_j}\frac{k}{\omega^2} + C_\mu k_a k - \frac{\partial u_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{k}{\omega^2} = 0\end{aligned} \tag{B.46}$$

The terms in the boundary integral will be used for the derivation of the boundary conditions.

$$\begin{aligned}D_i^u = &u_i v_j n_j + u_j v_j n_i + (\nu + \nu_t) n_j \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - q n_i + \omega_a \omega n_i - 2\omega_a \gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) n_j \\ &+ \frac{2}{3}\gamma \omega_a \omega n_i + k_a k n_i - 2k_a n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \frac{2}{3}k_a k n_i\end{aligned} \tag{B.47}$$

$$D^q = u_i n_i \tag{B.48}$$

$$D^{k_a} = -u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{1}{\omega} - \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{1}{\omega} + k_a v_j n_j - k_a n_j \sigma_k \frac{\partial k}{\partial x_j}\frac{1}{\omega} + \frac{\partial k_a}{\partial x_j}(\sigma_k \nu_t + \nu) n_j \tag{B.49}$$

$$D^{\omega_a} = u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{k}{\omega^2} + \omega_a n_j v_j + \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j}\frac{k}{\omega^2} + \frac{\partial \omega_a}{\partial x_j}(\sigma_\omega \nu_t + \nu) n_j + k_a n_j \sigma_k \frac{\partial k}{\partial x_j}\frac{k}{\omega^2} \tag{B.50}$$

$$P_{ij}^u = -u_i n_j - u_j n_i \tag{B.51}$$

$$P_j^k = -k_a(\sigma_k \nu_t + \nu) n_j \tag{B.52}$$

$$P_j^\omega = -\omega_a(\sigma_\omega \nu_t + \nu) n_j \tag{B.53}$$

B.2. Boundary conditions

B.2.1. Inlet and wall

On the inlet and no-slip walls, Dirichlet boundary conditions are applied on the primal variables, i.e.

$$\frac{\delta v_i}{\delta \beta} = 0, \qquad \frac{\delta k}{\delta \beta} = 0, \qquad \text{and} \qquad \frac{\delta \omega}{\delta \beta} = 0. \tag{B.54}$$

Therefore, in order to zero the remaining boundary integrals, impose

$$D^q = 0, \qquad P^u_{ij} = 0, \qquad P^k_j = 0, \qquad \text{and} \qquad P^\omega_j = 0. \tag{B.55}$$

From this follows,

$$D^q = u_i n_i = u_n = 0, \tag{B.56}$$

$$n_i P^u_{ij} = -u_i n_j n_i - u_j n_i n_i = -u_n n_j - u_j = -u_j = 0. \tag{B.57}$$

Therefore, all components of the adjoint velocity are zero. The remaining expressions give

$$\begin{aligned} P^k_j = -k_a(\sigma_k \nu_t + \nu) n_j = 0 \\ k_a = 0, \end{aligned} \tag{B.58}$$

and

$$\begin{aligned} P^\omega_j = -\omega_a(\sigma_\omega \nu_t + \nu) = 0 \\ \omega_a = 0. \end{aligned} \tag{B.59}$$

Therefore, zero Dirichlet boundary conditions are applied on the adjoint velocity, k_a, and ω_a. The inlet and wall boundary conditions for the adjoint pressure do not follow from these equations. [172] suggests to use a zero Neumann boundary condition.

B.2.2. Outlet

On the outlet, zero Neumann boundary conditions are applied on the primal velocity and the primal turbulent variables, and a zero Dirichlet boundary condition is imposed on the primal pressure.

$$\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta \beta}\right) = 0, \qquad \frac{\delta p}{\delta \beta} = 0, \qquad \frac{\partial}{\partial x_j}\left(\frac{\delta k}{\delta \beta}\right) = 0, \qquad \text{and} \qquad \frac{\partial}{\partial x_j}\left(\frac{\delta \omega}{\delta \beta}\right) = 0. \tag{B.60}$$

Again, to zero the remaining boundary terms, impose

$$D^u_i = 0, \qquad D^{k_a} = 0, \qquad \text{and} \qquad D^{\omega_a} = 0. \tag{B.61}$$

The boundary condition for the adjoint pressure can be derived from the normal component of the expression for D^u_i.

$$\begin{aligned} D^u_i n_i = u_n v_n + u_j v_j + (\nu + \nu_t) n_i n_j \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - q + \omega_a \omega - 2\omega_a \beta\gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) n_i n_j + \frac{2}{3}\gamma \omega_a \omega + k_a k \\ - 2k_a n_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \frac{2}{3} k_a k = 0 \\ q = u_n v_n + u_j v_j + 2(\nu + \nu_t) n_j \frac{\partial u_n}{\partial x_j} + \omega_a \omega - 2\omega_a \beta\gamma \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) n_i n_j + \frac{2}{3}\gamma \omega_a \omega + k_a k \\ - 2k_a n_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \frac{2}{3} k_a k \end{aligned} \tag{B.62}$$

Similarly, for the tangential component,

$$\begin{aligned} D_i^u - D_k^u n_i n_k = {} & u_i v_j n_j - u_k n_k v_j n_j n_i + (\nu + \nu_t) n_j \left(\frac{\partial}{\partial x_j} (u_i - u_k n_k n_i) + \frac{\partial u_j}{\partial x_i} - \frac{\partial u_j}{\partial x_k} n_k n_i \right) \\ & - 2\omega_a \beta\gamma n_j \left(\frac{\partial}{\partial x_j} (v_i - v_k n_k n_i) + \frac{\partial v_j}{\partial x_i} - \frac{\partial v_j}{\partial x_k} n_k n_i \right) \\ & - 2k_a n_j \left(\frac{\partial}{\partial x_j} (v_i - v_k n_k n_i) + \frac{\partial v_j}{\partial x_i} - \frac{\partial v_j}{\partial x_k} n_k n_i \right) \\ = {} & v_n u_{\parallel i} + (\nu + \nu_t) n_j \left(\frac{\partial u_{\parallel_i}}{\partial x_j} + \frac{\partial u_j}{\partial x_{\parallel i}} \right) \\ & - 2\omega_a \beta\gamma n_j \left(\frac{\partial v_{\parallel i}}{\partial x_j} + \frac{\partial v_j}{\partial x_{\parallel i}} \right) - 2k_a n_j \left(\frac{\partial v_{\parallel i}}{\partial x_j} + \frac{\partial v_j}{\partial x_{\parallel i}} \right). \end{aligned} \tag{B.63}$$

The boundary conditions for the adjoint turbulent variables can be derived from the last two boundary terms.

$$\begin{aligned} D^{k_a} &= -u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \frac{1}{\omega} - \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{1}{\omega} + k_a v_j n_j - k_a n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{1}{\omega} + \frac{\partial k_a}{\partial x_j} (\sigma_k \nu_t + \nu) n_j = 0 \\ \omega_a &= \frac{-u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \frac{1}{\omega} + k_a v_j n_j - k_a n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{1}{\omega} + \frac{\partial k_a}{\partial x_j} (\sigma_k \nu_t + \nu) n_j}{n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{1}{\omega}} \end{aligned} \tag{B.64}$$

$$\begin{aligned} D^{\omega_a} &= u_i n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \frac{k}{\omega^2} + \omega_a n_j v_j + \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{k}{\omega^2} + \frac{\partial \omega_a}{\partial x_j} (\sigma_\omega \nu_t + \nu) n_j + k_a n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{k}{\omega^2} = 0 \\ k_a &= \frac{-u_i n_j \left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i} \right) \frac{k}{\omega^2} - \omega_a n_j v_j - \omega_a n_j \sigma_\omega \frac{\partial \omega}{\partial x_j} \frac{k}{\omega^2} - \frac{\partial \omega_a}{\partial x_j} (\sigma_\omega \nu_t + \nu) n_j}{n_j \sigma_k \frac{\partial k}{\partial x_j} \frac{k}{\omega^2}} \end{aligned} \tag{B.65}$$

B.2.3. Slip wall

On a slip wall, the normal component of the primal velocity is zero, and zero Neumann boundary conditions are applied to the primal pressure and the turbulence variables, resulting in

$$\frac{\delta (n_j v_j)}{\delta \beta} = 0, \qquad \frac{\partial}{\partial x_j} \left(\frac{\delta k}{\delta \beta} \right), \qquad \text{and} \qquad \frac{\partial}{\partial x_j} \left(\frac{\delta \omega}{\delta \beta} \right), \tag{B.66}$$

As only the normal component of the velocity is zero at the wall, it is still necessary to set D_i^u to zero. Therefore, impose

$$D_i^u = 0, \qquad D^q = 0, \qquad D^{k_a} = 0, \qquad D^{\omega_a} = 0, \qquad \text{and} \qquad P_{ij}^u = 0 \tag{B.67}$$

From D^q and P_{ij}^u, it follows that

$$\begin{aligned} P_{ij}^u &= -u_i n_j - u_j n_i = 0 \\ n_i P_{ij}^u &= -u_i n_i n_j - u_j = 0 \\ &= -D^q n_j - u_j = 0 \\ &= -u_j = 0. \end{aligned} \tag{B.68}$$

Therefore, the adjoint velocity is zero on the slip wall, and the adjoint pressure and turbulence variables are determined in the same way as for the outlet.

B.3. Objective function

The objective function derived from the maximum a posteriori is given by

$$J = \frac{1}{2} (\mathbf{d}_{\text{RANS}} - \mathbf{d}_{\text{HF}})^T C_m^{-1} (\mathbf{d}_{\text{RANS}} - \mathbf{d}_{\text{HF}}) + \frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}})^T C_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}}). \tag{B.69}$$

In the simple case where a two single scalar variances are used for a diagonal observational and prior covariance matrix,

$$C_m = \sigma_m^2 I \qquad \text{and} \qquad C_\beta = \sigma_\beta^2 I, \tag{B.70}$$

the objective function then reduces to

$$J = \frac{1}{2\sigma_m^2}(\mathbf{d}_{\text{RANS}} - \mathbf{d}_{\text{HF}})^T(\mathbf{d}_{\text{RANS}} - \mathbf{d}_{\text{HF}}) + \frac{1}{2\sigma_\beta^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}})^T(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{prior}}) \tag{B.71}$$

$$= \frac{1}{2\sigma_m^2}\sum_{i=1}^{N_d}(d_{\text{RANS},i} - d_{\text{HF},i})^2 + \frac{1}{2\sigma_\beta^2}\sum_{j=1}^{N}(\beta_j - \beta_{\text{prior},j})^2, \tag{B.72}$$

such that

$$J_\Omega = \frac{1}{2\sigma_m^2}\sum_{i=1}^{N_d}\delta(x-x_i)(d_{\text{RANS},i} - d_{\text{HF},i})^2 + \frac{1}{2\sigma_\beta^2}\sum_{j=1}^{N}\delta(x-x_j)(\beta_j - \beta_{\text{prior},j})^2. \tag{B.73}$$

As explained in section 3.5.3, the Dirac delta functions are included because the data is only known at discrete locations (in this case the cell centres) while the objective function is still treated in a continuous manner. The Dirac delta functions disappear when the expressions are integrated. For the first term, this happens in the adjoint momentum equation. For the second term this is done in the derivation, such that only the sum over the grid cells needs to be implemented in the solver.

B.4. Production term correction

In this approach, the corrective term is applied to the production term in the ω equation, i.e.

$$R^\omega = \frac{\partial(v_j\omega)}{\partial x_j} - \frac{\partial}{\partial x_j}\left((\sigma_\omega \nu_t + \nu)\frac{\partial\omega}{\partial x_j}\right) - \beta\gamma P + \frac{2}{3}\gamma\frac{\partial v_j}{\partial x_j}\omega + \alpha\omega^2 = 0. \tag{B.74}$$

The required changes to the adjoint equation and the adjoint gradient can be derived from the production term.

$$\begin{aligned}
-\beta\gamma P &= -\gamma\beta\frac{\partial v_j}{\partial x_i}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \\
\frac{\delta(-\beta\gamma P)}{\delta\beta} &= -\gamma\frac{\partial v_j}{\partial x_i}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - 2\gamma\beta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) \\
\int_\Omega \omega_a(-\beta\gamma P)\,\mathrm{d}\Omega &= -\int_\Omega \omega_a\gamma\frac{\partial v_j}{\partial x_i}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\mathrm{d}\Omega - \int_\Omega 2\omega_a\gamma\beta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\mathrm{d}\Omega \\
&= -\int_\Omega \omega_a\gamma\frac{\partial v_j}{\partial x_i}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\mathrm{d}\Omega - \int_\Gamma 2\omega_a\gamma\beta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)n_j\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Gamma \\
&\quad + \int_\Omega \frac{\partial}{\partial x_j}\left(2\omega_a\gamma\beta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right)\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Omega
\end{aligned} \tag{B.75}$$

Collecting the terms, the adjoint gradient for this formulation is thus given by

$$\frac{\delta L}{\delta\beta} = -\int_\Omega \omega_a\gamma P\,\mathrm{d}\Omega, \tag{B.76}$$

excluding the term originating from the prior, which is the same for all corrective term formulations. Furthermore, the corrective terms should be included in the adjoint momentum equation and the adjoint boundary conditions corresponding to the latter two terms of (B.75).

B.5. Eigenvalue corrections

The eigendecomposition of the anisotropy tensor is given by

$$b = V\Lambda V^T, \tag{B.77}$$

where Λ is a matrix with the eigenvalues on the diagonal and V is a matrix of which the columns are the corresponding eigenvectors. The perturbed anisotropy tensor is given by

$$b' = V(\Lambda + B)V^T = V\Lambda V^T + VBV^T = b + VBV^T, \tag{B.78}$$

where

$$B = \text{diag}\,(\Delta\lambda_1, \Delta\lambda_2, \Delta\lambda_3) = \text{diag}\left(\frac{2}{3}\beta_{C_{1c}} + \frac{1}{6}\beta_{C_{2c}}, -\frac{1}{3}\beta_{C_{1c}} + \frac{1}{6}\beta_{C_{2c}}, -\frac{1}{3}\beta_{C_{1c}} - \frac{1}{3}\beta_{C_{2c}}\right). \tag{B.79}$$

The derivative of this matrix with respect to the corrective terms is then given by

$$\frac{\delta B}{\delta \beta_{C_{1c}}} = \text{diag}\left(\frac{2}{3}, -\frac{1}{3}, -\frac{1}{3}\right), \tag{B.80}$$

and

$$\frac{\delta B}{\delta \beta_{C_{1c}}} = \text{diag}\left(\frac{1}{6}, \frac{1}{6}, -\frac{1}{3}\right). \tag{B.81}$$

Looking at (B.78), this formulation has the convenient property that the unperturbed anisotropy tensor can be absorbed into the original expression of the momentum and k-equation. The correction can thus simply be added as a source term in both equations. Before deriving the relevant expressions, it can first be shown that the corrected turbulent kinetic energy is equal to the baseline turbulent kinetic energy.

$$k' = \frac{1}{2}\text{tr}\left(R'_{ij}\right) = k + k\,\text{tr}\left(VBV^T\right) = k + k\,\text{tr}\left(V^TVB\right) = k + k\,\text{tr}(B) = k \tag{B.82}$$

This follows from the fact that $b_{ii} = 0$ by definition and the trace is invariant under cyclic permutations. As b_{ij} is a real symmetric matrix, V is an orthogonal matrix and thus $V^TV = I$. The trace of B is zero by inferring only two of the barycentric coordinates and calculating the third. From the corrected anisotropy tensor, it can be derived that the corrected momentum equation is given by

$$R_i^{v'} = \underbrace{v_j\frac{\partial v_i}{\partial x_j} + \frac{\partial p}{\partial x_i} + \frac{2}{3}\frac{\partial k}{\partial x_i} - \frac{\partial}{\partial x_j}\left((\nu + \nu_t)\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right)}_{R_i^v} + \frac{\partial(2kV_{in}B_{nl}V_{jl})}{\partial x_j} = 0. \tag{B.83}$$

Similarly, the corrected k-equation is given by

$$R^{k'} = \frac{\partial(v_j k)}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\left(\sigma^*\nu_t + \nu\right)\frac{\partial k}{\partial x_j}\right) - \nu_t\frac{\partial v_i}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + 2k\frac{\partial v_i}{\partial x_j}V_{in}B_{nl}V_{jl} + C_\mu\omega k = 0. \tag{B.84}$$

For the momentum equation, the adjoint terms resulting from the additional source term can be derived as follows.

$$\begin{aligned}
\frac{\delta}{\delta\beta}\left(2\frac{\partial}{\partial x_j}\left(kV_{in}B_{nl}V_{jl}\right)\right) &= 2\frac{\partial}{\partial x_j}\left(V_{in}B_{nl}V_{jl}\frac{\delta k}{\delta\beta}\right) + 2\frac{\partial}{\partial x_j}\left(k\frac{\delta V_{in}B_{nl}V_{jl}}{\delta\beta}\right) \\
\int_\Omega u_i\frac{\delta}{\delta\beta}\left((2\frac{\partial}{\partial x_j}\left(kV_{in}B_{nl}V_{jl}\right)\right)\mathrm{d}\Omega &= \int_\Omega 2u_i\frac{\partial}{\partial x_j}\left(V_{in}B_{nl}V_{jl}\frac{\delta k}{\delta\beta}\right)\mathrm{d}\Omega + \int_\Omega 2u_i\frac{\partial}{\partial x_j}\left(kV_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\right)\mathrm{d}\Omega \\
&= \int_\Gamma 2u_in_jV_{in}B_{nl}V_{jl}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega 2\frac{\partial u_i}{\partial x_j}V_{in}B_{nl}V_{jl}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Omega \\
&\quad + \int_\Omega 2u_i\frac{\partial}{\partial x_j}\left(kV_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\right)\mathrm{d}\Omega
\end{aligned} \tag{B.85}$$

Similarly, for the k-equation:

$$\begin{aligned}
\frac{\delta}{\delta\beta}\left(2kV_{in}B_{nl}V_{jl}\frac{\partial v_i}{\partial x_j}\right) &= 2V_{in}B_{nl}V_{jl}\frac{\partial v_i}{\partial x_j}\frac{\delta k}{\delta\beta} + 2k\frac{\partial v_i}{\partial x_j}V_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl} + 2kV_{in}B_{nl}V_{jl}\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) \\
\int_\Omega k_a\frac{\delta R^k}{\delta\beta}\,\mathrm{d}\Omega &= \int_\Omega 2k_aV_{in}B_{nl}V_{jl}\frac{\partial v_i}{\partial x_j}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Omega + \int_\Omega 2k_ak\frac{\partial v_i}{\partial x_j}V_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\,\mathrm{d}\Omega \\
&\quad + \int_\Omega 2k_akV_{in}B_{nl}V_{jl}\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right)\mathrm{d}\Omega \\
&= \int_\Omega 2k_aV_{in}B_{nl}V_{jl}\frac{\partial v_i}{\partial x_j}\frac{\delta k}{\delta\beta}\,\mathrm{d}\Omega + \int_\Omega 2k_ak\frac{\partial v_i}{\partial x_j}V_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\,\mathrm{d}\Omega \\
&\quad + \int_\Gamma 2k_akV_{in}B_{nl}V_{jl}n_j\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega\frac{\partial}{\partial x_j}\left(2k_akV_{in}B_{nl}V_{jl}\right)\frac{\delta v_i}{\delta\beta}\,\mathrm{d}\Omega
\end{aligned} \tag{B.86}$$

Collecting the terms not including the sensitivities of any primal variables, the adjoint gradient is given by

$$\frac{\delta L}{\delta\beta} = \int_\Omega 2u_i\frac{\partial}{\partial x_j}\left(kV_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\right)\mathrm{d}\Omega + \int_\Omega 2k_ak\frac{\partial v_i}{\partial x_j}V_{in}\frac{\delta B_{nl}}{\delta\beta}V_{jl}\,\mathrm{d}\Omega + \frac{1}{\sigma_\beta^2}(\beta-\beta_{\text{prior}}). \tag{B.87}$$

B.6. Complete Reynolds stress tensor correction

B.6.1. Overview

When both the eigenvalues and the eigenvectors are corrected, the perturbed anisotropy tensor is given by

$$b' = QV(\Lambda+B)V^TQ^T, \tag{B.88}$$

where B is a diagonal matrix containing the respective corrections to the eigenvalues, as introduced in the previous section, and Q is the rotation matrix corresponding to the corrective rotation of the set of eigenvectors. If unit quaternions are used to represent this rotation, this matrix can be written as

$$Q = \begin{bmatrix} 1-2(q_j^2+q_k^2) & 2(q_iq_j-q_kq_r) & 2(q_iq_k+q_jq_r) \\ 2(q_iq_j+q_kq_r) & 1-2(q_i^2+q_k^2) & 2(q_jq_k-q_iq_r) \\ 2(q_iq_k-q_jq_r) & 2(q_jq_k+q_iq_r) & 1-2(q_i^2+q_j^2) \end{bmatrix}. \tag{B.89}$$

Three additional corrective terms then need to be formulated: β_{n_1} and β_{n_2} specifying the axis of the rotation, and β_θ specifying the rotation angle. Then, the derivatives of Q with respect to the rotation of the eigenvectors can be written as

$$\begin{aligned}
\frac{\delta q_{11}}{\delta\beta_\theta} &= -2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\left({n_2}^2+{n_3}^2\right) \\
\frac{\delta q_{12}}{\delta\beta_\theta} &= 2n_1\sin\frac{\theta}{2}n_2\cos\frac{\theta}{2} - 2n_3\cos^2\frac{\theta}{2} + n_3 \\
\frac{\delta q_{13}}{\delta\beta_\theta} &= 2n_1\sin\frac{\theta}{2}n_3\cos\frac{\theta}{2} + 2n_2\cos^2\frac{\theta}{2} - n_2 \\
\frac{\delta q_{21}}{\delta\beta_\theta} &= 2n_1\sin\frac{\theta}{2}n_2\cos\frac{\theta}{2} - 2n_3\cos^2\frac{\theta}{2} + n_3 \\
\frac{\delta q_{22}}{\delta\beta_\theta} &= -2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\left({n_1}^2+{n_3}^2\right) \\
\frac{\delta q_{23}}{\delta\beta_\theta} &= 2n_2\sin\frac{\theta}{2}n_3\cos\frac{\theta}{2} - 2n_1\cos^2\frac{\theta}{2} + n_1 \\
\frac{\delta q_{31}}{\delta\beta_\theta} &= 2n_1\sin\frac{\theta}{2}n_3\cos\frac{\theta}{2} + 2n_2\cos^2\frac{\theta}{2} - n_2 \\
\frac{\delta q_{32}}{\delta\beta_\theta} &= 2n_2\sin\frac{\theta}{2}n_3\cos\frac{\theta}{2} - 2n_1\cos^2\frac{\theta}{2} + n_1 \\
\frac{\delta q_{33}}{\delta\beta_\theta} &= -2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\left({n_1}^2+{n_2}^2\right)
\end{aligned} \tag{B.90}$$

And similarly for the rotation axis:

$$\frac{\delta Q}{\delta\beta_{n_1}} = \begin{bmatrix} 0 & 2n_2\sin^2\frac{\theta}{2} & 2n_3\sin^2\frac{\theta}{2} \\ 2n_2\sin^2\frac{\theta}{2} & -4n_1\sin^2\frac{\theta}{2} & -2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \\ 2n_2\sin^2\frac{\theta}{2} & -2\sin\frac{\theta}{2}\cos\frac{\theta}{2} & -4n_1\sin^2\frac{\theta}{2} \end{bmatrix}$$

and

$$\frac{\delta Q}{\delta\beta_{n_2}} = \begin{bmatrix} -4n_2\sin^2\frac{\theta}{2} & 2n_1\sin^2\frac{\theta}{2} & 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \\ 2n_1\sin^2\frac{\theta}{2} & 0 & 2n_3\sin^2\frac{\theta}{2} \\ 2n_1\sin^2\frac{\theta}{2} & -2n_3\sin^2\frac{\theta}{2} & -4n_2\sin^2\frac{\theta}{2} \end{bmatrix}$$

The momentum equation with the perturbed anisotropy tensor can be written as

$$R_i^v = \frac{\partial v_i v_j}{\partial x_j} + \frac{\partial p}{\partial x_i} + \frac{2}{3}\frac{\partial k}{\partial x_i} - \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) + \frac{\partial}{\partial x_j}\left(2k(QV(\Lambda+B)V^TQ^T)_{ij}\right) = 0. \tag{B.91}$$

The main difficulty in this derivation is that in this case the anisotropy tensor and its correction cannot be separated. Therefore, the complete corrected anisotropy tensor must be formulated as a source term in the momentum and k-equation. Again, the required changes to the adjoint equations and the expression for the gradient can be derived from the last term.

$$\begin{aligned}
\frac{\delta}{\delta\beta}\left(\frac{\partial}{\partial x_j}\left(2kb'_{ij}\right)\right) &= \frac{\partial}{\partial x_j}\left(2QV(\Lambda+B)V^TQ^T\frac{\delta k}{\delta\beta}\right) + \frac{\partial}{\partial x_j}\left(2k\frac{\delta}{\delta\beta}\left(QV(\Lambda+B)V^TQ^T\right)\right)\\
\int_\Omega u_i\frac{\delta}{\delta\beta}\left(\frac{\partial}{\partial x_j}\left(2kb'_{ij}\right)\right)\mathrm{d}\Omega &= \int_\Omega u_i\frac{\partial}{\partial x_j}\left(2QV(\Lambda+B)V^TQ^T\frac{\delta k}{\delta\beta}\right)\mathrm{d}\Omega\\
&\quad + \int_\Omega u_i\frac{\partial}{\partial x_j}\left(2k\frac{\delta}{\delta\beta}\left(QV(\Lambda+B)V^TQ^T\right)\right)\mathrm{d}\Omega\\
&= \int_\Gamma 2u_in_jQV(\Lambda+B)V^TQ^T\frac{\delta k}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega 2\frac{\partial u_i}{\partial x_j}QV(\Lambda+B)V^TQ^T\frac{\delta k}{\delta\beta}\,\mathrm{d}\Omega\\
&\quad + \int_\Omega u_i\frac{\partial}{\partial x_j}\left(2k\left(\frac{\delta Q}{\delta\beta}V(\Lambda+B)V^TQ^T + QV(\Lambda+B)V^T\frac{\delta Q^T}{\delta\beta} + QV\frac{\delta B}{\delta\beta}V^TQ^T\right)_{ij}\right)\mathrm{d}\Omega\\
&\quad \underbrace{+\int_\Omega u_i\frac{\partial}{\partial x_j}\left(2k\left(QV\frac{\delta\Lambda}{\delta\beta}V^TQ^T\right)_{ij}\right)\mathrm{d}\Omega}_{\mathrm{I}}\\
&\quad \underbrace{+\int_\Omega u_i\frac{\partial}{\partial x_j}\left(2k\left(Q\frac{\delta V}{\delta\beta}(\Lambda+B)V^TQ^T\right)_{ij}\right)\mathrm{d}\Omega}_{\mathrm{II}}\\
&\quad \underbrace{+\int_\Omega u_i\frac{\partial}{\partial x_j}\left(2k\left(QV(\Lambda+B)\frac{\delta V^T}{\delta\beta}Q^T\right)_{ij}\right)\mathrm{d}\Omega}_{\mathrm{III}}
\end{aligned} \tag{B.92}$$

The first two terms contribute to the adjoint k boundary condition and the k_a-equation, respectively. The third term is the expression for the adjoint gradient for this particular corrective formulation. In order to complete the derivation, it is necessary to expand the last three terms such that they can be expressed as boundary and domain integrals of expressions containing sensitivities with respect to the primal variables. The sensitivity of the unperturbed anisotropy tensor can be straightforwardly derived to be

$$\frac{\delta b_{ij}}{\delta\beta} = \frac{1}{2\omega^2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\delta\omega}{\delta\beta} - \frac{1}{2\omega}\left(\frac{\partial}{\partial x_j}\left(\frac{\delta v_i}{\delta\beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta v_j}{\delta\beta}\right)\right). \tag{B.93}$$

Therefore, if it is possible to find the sensitivities of the eigenvalues and eigenvectors in terms of the sensitivity of the anisotropy tensor, the derivation can be completed.

B.6.2. Eigenvalue derivatives

The derivation in this section follows the work of Fox and Kapoor [37], Nelson [97]. For the α-th eigenvalue and eigenvector, the following is true:

$$bv_\alpha = \lambda_\alpha v_\alpha. \tag{B.94}$$

Considering the derivative with respect to the corrective functions,

$$\frac{\delta b}{\delta\beta}v_\alpha + b\frac{\delta v_\alpha}{\delta\beta} = \frac{\delta\lambda_\alpha}{\delta\beta}v_\alpha + \lambda_\alpha\frac{\delta v_\alpha}{\delta\beta}. \tag{B.95}$$

Imposing that the eigenvectors have unit length gives $v_\alpha^T v_\alpha = 1$, and

$$\frac{\delta}{\delta\beta}\left(v_\alpha^\intercal v_\alpha\right) = 2v_{\alpha1}\frac{\delta v_{\alpha1}}{\delta\beta} + 2v_{\alpha2}\frac{\delta v_{\alpha2}}{\delta\beta} + 2v_{\alpha3}\frac{\delta v_{\alpha3}}{\delta\beta} = 2v_\alpha\cdot\frac{\delta v_\alpha}{\delta\beta} = 0, \tag{B.96}$$

as a variation in β should not change the length of the eigenvectors. Then, (B.95) simplifies to

$$v_\alpha^\intercal \frac{\delta b}{\delta \beta} v_\alpha + v_\alpha^\intercal b \frac{\delta v_\alpha}{\delta \beta} = \frac{\delta \lambda_\alpha}{\delta \beta}. \tag{B.97}$$

Also,

$$v_\alpha^\intercal b = v_\alpha^\intercal b^\intercal = (b v_\alpha)^\intercal = (\lambda_\alpha v_\alpha)^\intercal = \lambda_\alpha v_\alpha^\intercal. \tag{B.98}$$

Therefore,

$$\frac{\delta \lambda_\alpha}{\delta \beta} = v_\alpha^\intercal \frac{\delta b}{\delta \beta} v_\alpha. \tag{B.99}$$

B.6.3. Eigenvector derivatives

In order to derive the derivatives of the eigenvectors, define

$$F_p \equiv b - \lambda_p I. \tag{B.100}$$

Then,

$$F_p v_p = b v_p - \lambda_p I v_p = b v_p - \lambda_p v_p = 0. \tag{B.101}$$

Taking the derivative,

$$F_p \frac{\delta v_p}{\delta \beta} = -\frac{\delta F_p}{\delta \beta} v_p. \tag{B.102}$$

The combination of this expression with (B.96) gives

$$\begin{aligned}
\begin{bmatrix} F_p \\ \hdashline 2v_p^T \end{bmatrix} \frac{\delta v_p}{\delta \beta} &= \begin{bmatrix} -\frac{\delta F_p}{\delta \beta} \\ \hdashline 0 \end{bmatrix} v_p \\
\begin{bmatrix} F_p & \vdots & v_p \end{bmatrix} \begin{bmatrix} F_p \\ \hdashline 2v_p^T \end{bmatrix} \frac{\delta v_p}{\delta \beta} &= \begin{bmatrix} F_p & \vdots & v_p \end{bmatrix} \begin{bmatrix} -\frac{\delta F_p}{\delta \beta} \\ \hdashline 0 \end{bmatrix} v_p \\
\left[F_p F_p + 2 v_p v_p^T \right] \frac{\delta v_p}{\delta \beta} &= -F_p \frac{\delta F_p}{\delta \beta} v_p \\
\frac{\delta v_p}{\delta \beta} &= -\left[F_p F_p + 2 v_p v_p^T \right]^{-1} F_p \frac{\delta F_p}{\delta \beta} v_p
\end{aligned} \tag{B.103}$$

These expressions can then be used in (B.92), after which the derivation of the adjoint equations can be completed in a manner similar to the other formulations. This derivation has yet to be implemented and tested in OpenFOAM.

C

OpenFOAM implementation

This appendix gives the essential code for the implementation of the paradigm in OpenFOAM. The primal and adjoint implementation of the SIMPLE algorithm is given in listing 1. The relevant sections of the code are annotated. A more elaborate explanation of its workings is given in section 6.3.4 and fig. 6.1. The adjoint momentum equation is implemented in listing 2. The primal and adjoint equations for k and ω are implemented in `adjointkOmega`. The equations for k_a and ω_a are shown in listing 3.

```
#include "IOmanip.H"

#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "turbulentTransportModel.H"
#include "simpleControl.H"
#include "fvOptions.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "postProcess.H"

    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "createControl.H"
    #include "createFields.H"
    #include "createFvOptions.H"
    #include "initContinuityErrs.H"

    turbulence->validate();

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nStarting time loop\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        // Calculate the eigenvectors from the anisotropy tensor (this only done when the eigenvalues are
        ↪  corrected)
        #include "eig.H"

        // Solve the primal velocity and pressure
        {
            #include "UEqn.H"
            #include "pEqn.H"
        }

        // Solve the adjoint velocity and pressure
```

```
43          {
44              #include "UaEqn.H"
45              #include "paEqn.H"
46          }
47
48          laminarTransport.correct();
49
50          // Solve the primal and adjoint k and omega (this is done in adjointkOmega.C
51          turbulence->correct();
52
53          // Calculate the objective function
54          #include "calcJ.H"
55          Info << setprecision(15);
56          Info << "J: " << J << endl;
57
58          // Get the adjoint gradients from the adjoint turbulence model
59          volScalarField gradAdjointMag = turbulence->db().objectRegistry::lookupObject<volScalarField>
             ↪  ("gradAdjointMag");
60          volScalarField gradAdjointC1c = turbulence->db().objectRegistry::lookupObject<volScalarField>
             ↪  ("gradAdjointC1c");
61          volScalarField gradAdjointC2c = turbulence->db().objectRegistry::lookupObject<volScalarField>
             ↪  ("gradAdjointC2c");
62
63          // Add the contributions from the prior
64          gradAdjointMag += 1.0/sqr(sigma_betaCorrMag)*betaCorrMag;
65          gradAdjointC1c += 1.0/sqr(sigma_betaCorrC1c)*betaCorrC1c;
66          gradAdjointC2c += 1.0/sqr(sigma_betaCorrC2c)*betaCorrC2c;
67
68          // Gradient descent step (one-shot implementation)
69          betaCorrMag -= gradAdjointMag*lrMag;
70          betaCorrC1c -= gradAdjointC1c*lrC1c;
71          betaCorrC2c -= gradAdjointC2c*lrC2c;
72
73          runTime.write();
74
75          Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
76              << "  ClockTime = " << runTime.elapsedClockTime() << " s"
77              << nl << endl;
78      }
79
80      Info<< "End\n" << endl;
81
82      return 0;
83  }
```

Listing 1: Primal and adjoint SIMPLE algorithm - `adjointSimpleFoam.C`.

```
volVectorField dataTerm(1.0/sqr(sigma_m)*(U-UHF));

forAll(dataTerm, i)
{
    dataTerm[i].x() *= 1.0/mesh.V()[i];
    dataTerm[i].y() = 0.0;
    dataTerm[i].z() = 0.0;
}

dataTermTest = dataTerm;

tmp<fvVectorMatrix> tUaEqn
(
    fvm::div(-phi, Ua)
  - (U & fvc::grad(Ua))
  + turbulence->divDevReff(Ua)
 ==
  - velocityTerm
  - fvc::div(velocityTermLapl*dev(twoSymm(fvc::grad(U))))
  - dataTerm
  + fvOptions(Ua)
);

fvVectorMatrix& UaEqn = tUaEqn.ref();

UaEqn.relax();

fvOptions.constrain(UaEqn);

if (simple.momentumPredictor())
{
    solve(UaEqn == -fvc::grad(pa));

    fvOptions.correct(Ua);
}
```

Listing 2: Adjoint momentum equation - `UaEqn.H`.

```
tmp<fvScalarMatrix> omegaaEqn
(
    fvm::div(-alphaRhoPhi, omegaa_)
  - fvm::laplacian(alpha*rho*DomegaEff(), omegaa_)
 ==
    (fvc::grad(omegaa_) & fvc::grad(omega_))*alphaOmega_*k_/sqr(omega_)
  + (fvc::grad(ka_) & fvc::grad(k_))*alphaK_*k_/sqr(omega_)
  - fvm::SuSp(((2.0/3.0)*gamma_)*divU, omegaa_)
  + (fvc::grad(Ua) && dev(twoSymm(fvc::grad(U))))*k_/sqr(omega_)
  - ka_*P*k_/sqr(omega_)
  - Cmu_*ka_*k_
  - fvm::Sp(2.0*beta_*omega_, omegaa_)
  + fvOptions(alpha, rho, omegaa_)
);

omegaaEqn.ref().relax();
fvOptions.constrain(omegaaEqn.ref());
omegaaEqn.ref().boundaryManipulate(omegaa_.boundaryFieldRef());
solve(omegaaEqn);
fvOptions.correct(omegaa_);


tmp<fvScalarMatrix> kaEqn
(
    fvm::div(-alphaRhoPhi, ka_)
  - fvm::laplacian(alpha*rho*DkEff(), ka_)
 ==
  - (fvc::grad(ka_) & fvc::grad(k_))*alphaK_/omega_
  - (fvc::grad(omegaa_) & fvc::grad(omega_))*alphaOmega_/omega_
  + ka_*P/omega_
  - fvm::Sp(Cmu_*omega_, ka_)
  - (fvc::grad(Ua) && dev(twoSymm(fvc::grad(U))))/omega_
  - fvm::SuSp((2.0/3.0)*alpha*rho*divU, ka_)
  + fvOptions(alpha, rho, ka_)
);

kaEqn.ref().relax();
fvOptions.constrain(kaEqn.ref());
kaEqn.ref().boundaryManipulate(ka_.boundaryFieldRef());
solve(kaEqn);
fvOptions.correct(ka_);

velocityTerm_ = - omega_*fvc::grad(omegaa_)
                - k_*fvc::grad(ka_)
                - 2.0/3.0*gamma_*fvc::grad(omegaa_*omega_)
                - 2.0/3.0*fvc::grad(ka_*k_)
                ;

velocityTermLapl_ = 2.0*omegaa_*betaCorr_*gamma_ + 2.0*nut*ka_;
```

Listing 3: Adjoint $k-\omega$ model - `adjointkOmega.C`. For the sake of brevity, only the implementation for the production term correction is given.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] K. Abe, Y.-J. Jang, and M. A. Leschziner. An investigation of wall-anisotropy expressions and length-scale equations for non-linear eddy-viscosity models. *International Journal of Heat and Fluid Flow*, 24 (2):181–198, 2003.

[3] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4-5):443–480, 1999.

[4] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*. Elsevier, 2018.

[5] I. Azijli, A. Sciacchitano, D. Ragni, A. Palha, and R. P. Dwight. A posteriori uncertainty quantification of PIV-based pressure data. *Experiments in Fluids*, 57(5):72, 2016.

[6] H. S. Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[7] B. Baldwin and H. Lomax. Thin-layer approximation and algebraic model for separated turbulent flows. In *16th aerospace sciences meeting*, page 257, 1978.

[8] S. Banerjee, R. Krahl, F. Durst, and C. Zenger. Presentation of anisotropy properties of turbulence, invariants versus eigenvalue approaches. *Journal of Turbulence*, 8:N32, 2007.

[9] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: an introduction*, volume 1. Morgan Kaufmann San Francisco, 1998.

[10] O. Baysal and M. E. Eleshaky. Aerodynamic sensitivity analysis methods for the compressible Euler equations. *Journal of Fluids Engineering*, 113(4):681–688, 1991.

[11] M. Benning and M. Burger. Modern regularization methods for inverse problems. *Acta Numerica*, 27: 1–111, 2018.

[12] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[13] J. Borggaard, J. Burns, E. Cliff, and M. Gunzburger. Sensitivity calculations for a 2D, inviscid, supersonic forebody problem. *Identification and control of systems governed by partial differential equations*, pages 14–24, 1993.

[14] L. Breiman and P. Spector. Submodel selection and evaluation in regression. the x-random case. *International statistical review/revue internationale de Statistique*, pages 291–319, 1992.

[15] M. Breuer, N. Peller, C. Rapp, and M. Manhart. Flow over periodic hills–numerical and experimental study in a wide range of Reynolds numbers. *Computers & Fluids*, 38(2):433–457, 2009.

[16] A. Bueno-Orovio, C. Castro, F. Palacios, and E. Zuazua. Continuous adjoint approach for the Spalart-Allmaras model in aerodynamic optimization. *AIAA journal*, 50(3):631–646, 2012.

[17] A. Campos, K. Duraisamy, and G. Iaccarino. Towards a two-equation algebraic structure-based model with applications to turbulent separated flows. In *21st AIAA Computational Fluid Dynamics Conference*, page 2719, 2013.

[18] B. P. Carlin and T. A. Louis. *Bayesian methods for data analysis*. CRC Press, 2008.

[19] T. Cebeci and A. Smith. Analysis of turbulent boundary layers academic. *New York*, 1974.

[20] S. H. Cheung, T. A. Oliver, E. E. Prudencio, S. Prudhomme, and R. D. Moser. Bayesian uncertainty analysis with applications to turbulence modeling. *Reliability Engineering & System Safety*, 96(9):1137–1149, 2011.

[21] G. Chiandussi. Development of a shape optimization technique based on a response surface methodology. *Publication CIMNE*, (138), 1998.

[22] F. Chollet et al. Keras. `https://keras.io`, 2015.

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] T. Craft, B. Launder, and K. Suga. Development and application of a cubic eddy-viscosity model of turbulence. *International Journal of Heat and Fluid Flow*, 17(2):108–115, 1996.

[25] W. C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17, 1991.

[26] E. Dow and Q. Wang. Uncertainty quantification of structural uncertainties in RANS simulations of complex flows. In *20th AIAA Computational Fluid Dynamics Conference*, page 3865, 2011.

[27] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[28] K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *arXiv preprint arXiv:1804.00183*, 2018.

[29] K. Duraisamy, Z. J. Zhang, and A. P. Singh. New approaches in turbulence and transition modeling using data-driven techniques. In *53rd AIAA Aerospace Sciences Meeting*, page 1284, 2015.

[30] R. Dwight, J. Brezillon, and D. Vollmer. Efficient algorithms for solution of the adjoint compressible Navier-Stokes equations with applications. In *Proceedings of the ONERA-DLR aerospace symposium (ODAS), Toulouse*. Citeseer, 2006.

[31] R. Dwight and Z.-H. Han. Efficient uncertainty quantification using gradient-enhanced kriging. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No*, page 2276, 2009.

[32] W. Edeling, P. Cinnella, and R. P. Dwight. Predictive RANS simulations via Bayesian model-scenario averaging. *Journal of Computational Physics*, 275:65–91, 2014.

[33] W. Edeling, P. Cinnella, R. P. Dwight, and H. Bijl. Bayesian estimates of parameter variability in the $k-\varepsilon$ turbulence model. *Journal of Computational Physics*, 258:73–94, 2014.

[34] W. Edeling, G. Iaccarino, and P. Cinnella. A return to eddy viscosity model for epistemic UQ in RANS closures. *arXiv preprint arXiv:1705.05354*, 2017.

[35] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7 (2):149–154, 1964.

[36] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[37] R. Fox and M. Kapoor. Rates of change of eigenvalues and eigenvectors. *AIAA journal*, 6(12):2426–2429, 1968.

[38] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[39] J. Fröhlich, C. P. Mellen, W. Rodi, L. Temmerman, and M. A. Leschziner. Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions. *Journal of Fluid Mechanics*, 526:19–66, 2005.

[40] P. Gage and I. Kroo. A role for genetic algorithms in a preliminary design environment. In *Aircraft Design, Systems, and Operations Meeting*, page 3933, 1993.

[41] X. Gao, Y. Wang, N. Overton, M. Zupanski, and X. Tu. Data-assimilated computational fluid dynamics modeling of convection-diffusion-reaction problems. *Journal of computational science*, 21:38–59, 2017.

[42] M. Giles, N. Pierce, M. Giles, and N. Pierce. Adjoint equations in CFD-duality, boundary conditions and solution behaviour. In *13th Computational Fluid Dynamics Conference*, page 1850, 1997.

[43] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3-4):393–415, 2000.

[44] A. A. Giunta, J. M. Dudley, R. Narducci, B. Grossman, R. T. Haftka, W. H. Mason, and L. T. Watson. Noisy aerodynamic response and smooth approximations in HSCT design. In *Proc. 5-th AIAA/USAF/NASA/ISSMO Symp. on Multidisciplinary and Structural Optimization*, pages 1117–1128, 1994.

[45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[46] A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[47] K. Hanjalic. Will RANS survive LES? a view of perspectives. *Journal of fluids engineering*, 127(5):831–839, 2005.

[48] P. C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM review*, 34(4): 561–580, 1992.

[49] R. Hartmann. Adjoint consistency analysis of discontinuous galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[50] C. He, Y. Liu, and L. Gan. A data assimilation model for turbulent flows using continuous adjoint formulation. *Physics of Fluids*, 30(10):105108, 2018.

[51] T. Hermann. frugally-deep. `https://github.com/Dobiasd/frugally-deep`, 2018.

[52] J. R. Holland, J. D. Baeder, and K. Duraisamy. Towards integrated field inversion and machine learning with embedded neural networks for RANS modeling. In *AIAA Scitech 2019 Forum*, page 1884, 2019.

[53] S. Hoyas and J. Jiménez. Reynolds number effects on the Reynolds-stress budgets in turbulent channels. *Physics of Fluids*, 20(10):101511, 2008.

[54] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[55] G. Iaccarino, A. A. Mishra, and S. Ghili. Eigenspace perturbations for uncertainty estimation of single-point turbulence closures. *Physical Review Fluids*, 2(2):024605, 2017.

[56] M. A. Iglesias, K. J. Law, and A. M. Stuart. Ensemble Kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013.

[57] S. Jakirlić, R. Manceau, S. Šarić, A. Fadai-Ghotbi, B. Kniesner, S. Carpy, G. Kadavelil, C. Friess, C. Tropea, and J. Borée. LES, zonal and seamless hybrid LES/RANS: rationale and application to free and wall-bounded flows involving separation and swirl. In *Numerical simulation of turbulent flows and noise generation*, pages 253–282. Springer, 2009.

[58] S. Z. Jakirlic and R. Maduta. On "steady" RANS modeling for improved prediction of wall-bounded separation. In *52nd Aerospace Sciences Meeting*, page 0586, 2014.

[59] A. Jameson. Aerodynamic design via control theory. *Journal of scientific computing*, 3(3):233–260, 1988.

[60] A. Jameson. Optimum aerodynamic design via boundary control. *Research Institute for Advanced Computer Science, NASA Ames Research Center*, 1994.

[61] A. Jameson and J. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 9, 01 2000.

[62] A. Jameson, L. Martinelli, and N. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and computational fluid dynamics*, 10(1-4):213–237, 1998.

[63] H. Jasak, A. Jemcov, Z. Tukovic, et al. OpenFOAM: A C++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20. IUC Dubrovnik, Croatia, 2007.

[64] C. Johnson and R. Rannacher. On error control in CFD. In *Numerical methods for the Navier-Stokes equations*, pages 133–144. Springer, 1994.

[65] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`.

[66] S. Jovic and D. M. Driver. Backward-facing step measurements at low Reynolds number, $\mathrm{Re}_h = 5000$. *National Aeronautics and Space Administration, Ames Research Center*, 1994.

[67] M. Kaandorp. Machine learning for data-driven RANS turbulence modelling. Master's thesis, Delft University of Technology, 2018.

[68] M. L. Kaandorp and R. P. Dwight. Stochastic random forests with invariance for RANS turbulence modelling. *arXiv preprint arXiv:1810.08794*, 2018.

[69] H. Kato and S. Obayashi. Statistical approach for determining parameters of a turbulence model. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 2452–2457. IEEE, 2012.

[70] H. Kato, A. Yoshizawa, G. Ueno, and S. Obayashi. A data assimilation methodology for reconstructing turbulent flows around aircraft. *Journal of Computational Physics*, 283:559–581, 2015.

[71] I. Kavvadias, E. Papoutsis-Kiachagias, G. Dimitrakopoulos, and K. Giannakoglou. The continuous adjoint approach to the $k-\omega$ SST turbulence model with applications in shape optimization. *Engineering Optimization*, 47(11):1523–1542, 2015.

[72] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[73] K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In *Intelligent Control and Adaptive Systems*, volume 1196, pages 289–297. International Society for Optics and Photonics, 1990.

[74] B. E. Launder and B. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in heat and mass transfer*, 1(2):131–137, 1974.

[75] B. E. Launder, G. J. Reece, and W. Rodi. Progress in the development of a Reynolds-stress turbulence closure. *Journal of fluid mechanics*, 68(3):537–566, 1975.

[76] J.-P. Laval and M. Marquillie. Direct numerical simulations of converging–diverging channel flow. In *Progress in Wall Turbulence: Understanding and Modeling*, pages 203–209. Springer, 2011.

[77] H. Le, P. Moin, and J. Kim. Direct numerical simulation of turbulent flow over a backward-facing step. *Journal of fluid mechanics*, 330:349–374, 1997.

[78] S. Lefantzi, J. Ray, S. Arunajatesan, and L. Dechant. Tuning a RANS $k-\varepsilon$ model for jet-in-crossflow simulations. Technical report, Sandia National Laboratories (SNL-CA), Livermore, CA (United States); Sandia National Laboratories, Albuquerque, NM, 2014.

[79] M. Lefik and B. Schrefler. Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Computer methods in applied mechanics and engineering*, 192(28-30):3265–3283, 2003.

[80] J. Ling and J. Templeton. Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. *Physics of Fluids*, 27(8):085103, 2015.

[81] J. Ling, R. Jones, and J. Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

[82] J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[83] A. Lozano-Durán and J. Jiménez. Effect of the computational domain on direct simulations of turbulent channels up to $\mathrm{Re}_\tau = 4200$. *Physics of Fluids*, 26(1):011702, 2014.

[84] J. L. Lumley. Computational modeling of turbulent flows. In *Advances in applied mechanics*, volume 18, pages 123–176. Elsevier, 1979.

[85] I. Marusic, G. Candler, V. Interrante, P. Subbareddy, and A. Moss. Real time feature extraction for the analysis of turbulent flows. In *Data Mining for Scientific and Engineering Applications*, pages 223–238. Springer, 2001.

[86] C. Mellen, J. Fr, W. Rodi, et al. Large eddy simulations of the flow over periodic hills. 2000.

[87] F. R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, 32(8):1598–1605, 1994.

[88] F. Menter. Zonal two equation $k-\omega$ turbulence models for aerodynamic flows. In *23rd fluid dynamics, plasmadynamics, and lasers conference*, page 2906, 1993.

[89] F. Menter. Influence of freestream values on $k-\omega$ turbulence model predictions. *AIAA journal*, 30(6): 1657–1659, 1992.

[90] F. Menter and Y. Egorov. The scale-adaptive simulation method for unsteady turbulent flow predictions. part 1: theory and model description. *Flow, Turbulence and Combustion*, 85(1):113–138, 2010.

[91] F. Menter, A. Garbaruk, and Y. Egorov. Explicit algebraic Reynolds stress models for anisotropic wall-bounded flows. *Progress in Flight Physics*, 3:89–104, 2012.

[92] M. Milano and P. Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[93] R. D. Moser, J. Kim, and N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau = 590$. *Physics of fluids*, 11(4):943–945, 1999.

[94] K. P. Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

[95] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, page 667, 2000.

[96] R. Narducci, B. Grossman, M. Valorani, A. Dadone, and R. Haftka. Optimization methods for non-smooth or noisy objective functions in fluid design problems. In *12th Computational Fluid Dynamics Conference*, page 1648, 1995.

[97] R. B. Nelson. Simplified calculation of eigenvector derivatives. *AIAA journal*, 14(9):1201–1205, 1976.

[98] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006.

[99] S. Obayashi and T. Tsukahara. Comparison of optimization algorithms for aerodynamic shape design. *AIAA journal*, 35(8):1413–1415, 1997.

[100] T. Oliver and R. Moser. Uncertainty quantification for RANS turbulence model predictions. In *APS division of fluid dynamics meeting abstracts*, 2009.

[101] T. A. Oliver and R. D. Moser. Bayesian uncertainty quantification applied to RANS turbulence models. In *Journal of Physics: Conference Series*, volume 318, page 042032. IOP Publishing, 2011.

[102] C. Othmer, E. de Villiers, and H. Weller. Implementation of a continuous adjoint for topology optimization of ducted flows. In *18th AIAA Computational Fluid Dynamics Conference*, page 3947, 2007.

[103] D. Papadimitriou and K. Giannakoglou. Computation of the Hessian matrix in aerodynamic inverse design using continuous adjoint formulations. *Computers & Fluids*, 37(8):1029–1039, 2008.

[104] D. Papadimitriou and K. Giannakoglou. Direct, adjoint and mixed approaches for the computation of Hessian in airfoil design problems. *International journal for numerical methods in fluids*, 56(10): 1929–1943, 2008.

[105] E. Papoutsis-Kiachagias, A. Zymaris, I. Kavvadias, D. Papadimitriou, and K. Giannakoglou. The continuous adjoint approach to the $k-\varepsilon$ turbulence model for shape optimization and optimal active control of turbulent flows. *Engineering Optimization*, 47(3):370–389, 2015.

[106] E. Parish and K. Duraisamy. Quantification of turbulence modeling uncertainties using full field inversion. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2459, 2015.

[107] E. J. Parish and K. Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774, 2016.

[108] S. Patankar. *Numerical heat transfer and fluid flow*. CRC press, 1980.

[109] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[110] J. E. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[111] A. Pinelli, M. Uhlmann, A. Sekimoto, and G. Kawahara. Reynolds number dependence of mean flow structure in square duct turbulence. *Journal of fluid mechanics*, 644:107–122, 2010.

[112] M. Pini, P. Cinnella, et al. Hybrid adjoint-based robust optimization approach for fluid-dynamics problems. In *15th AIAA Non-Deterministic Approaches Conference*, page 16, 2012.

[113] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110, 1974.

[114] C. Poloni and G. Mosetti. Aerodynamic shape optimization by means of a genetic algorithm. In *the 5th international symposium on computational fluid dynamics*, pages 279–284. Japan Society of Computational Fluid Dynamics, 1993.

[115] S. Pope. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2):331–340, 1975.

[116] S. B. Pope. *Turbulent flows*. Cambridge Univ. Press, Cambridge, 2011.

[117] S. Poroseva and S. M. Murman. Velocity/pressure-gradient correlations in a FORANS approach to turbulence modeling. In *44th AIAA Fluid Dynamics Conference*, page 2207, 2014.

[118] S. V. Poroseva, J. D. Colmenares F, and S. M. Murman. On the accuracy of RANS simulations with DNS data. *Physics of Fluids*, 28(11):115102, 2016.

[119] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1): 145–151, 1999.

[120] D. Quagliarella and A. Della Cioppa. Genetic algorithms applied to the aerodynamic design of transonic airfoils. *Journal of aircraft*, 32(4):889–891, 1995.

[121] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams. Approximation methods for Gaussian process regression. *Large-scale kernel machines*, pages 203–224, 2007.

[122] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(Nov):2491–2521, 2008.

[123] L. B. Rall. *Automatic differentiation: Techniques and applications*. Springer, 1981.

[124] R. Rannacher and F.-T. Suttmeier. A posteriori error control in finite element methods via duality techniques: Application to perfect plasticity. *Computational mechanics*, 21(2):123–133, 1998.

[125] C. E. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[126] J. Ray, S. Lefantzi, S. Arunajatesan, and L. Dechant. Learning an eddy viscosity model using shrinkage and bayesian calibration: A jet-in-crossflow case study. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 4(1):011001, 2018.

[127] W. Rodi. A new algebraic relation for calculating the Reynolds stresses. In *Gesellschaft Angewandte Mathematik und Mechanik Workshop Paris France*, volume 56, 1976.

[128] J. Rotta. Statistische theorie nichthomogener turbulenz. *Zeitschrift für Physik*, 129(6):547–572, 1951.

[129] C. L. Rumsey. Exploring a method for improving turbulent separated-flow predictions with $k-\omega$ models. 2009.

[130] M. Schramm, B. Stoevesandt, and J. Peinke. Optimization of airfoils using the adjoint approach and the influence of adjoint turbulent viscosity. *Computation*, 6(1):5, 2018.

[131] U. Schumann. Realizability of Reynolds-stress turbulence models. *The Physics of Fluids*, 20(5):721–725, 1977.

[132] V. Sekar, M. Zhang, C. Shu, and B. C. Khoo. Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal*, pages 1–11, 2019.

[133] T. Shih, J. Zhu, and J. Lumley. A realizable Reynolds stress algebraic equation model NASA tm 105993. Technical report, ICOMP-92-27, CMOTT-92-14, 1992.

[134] R. L. Simpson. Turbulent boundary-layer separation. *Annual Review of Fluid Mechanics*, 21(1):205–232, 1989.

[135] A. P. Singh and K. Duraisamy. Using field inversion to quantify functional errors in turbulence closures. *Physics of Fluids*, 28(4):045110, 2016.

[136] A. P. Singh, K. Duraisamy, and Z. J. Zhang. Augmentation of turbulence models using field inversion and machine learning. In *55th AIAA Aerospace Sciences Meeting*, page 0993, 2017.

[137] A. P. Singh, R. Matai, A. Mishra, K. Duraisamy, and P. A. Durbin. Data-driven augmentation of turbulence models for adverse pressure gradient flows. In *23rd AIAA Computational Fluid Dynamics Conference*, page 3626, 2017.

[138] A. P. Singh, S. Medida, and K. Duraisamy. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, pages 1–13, 2017.

[139] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: a path to revolutionary computational aerosciences. 2014.

[140] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[141] C. G. Speziale, S. Sarkar, and T. B. Gatski. Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach. *Journal of fluid mechanics*, 227:245–272, 1991.

[142] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM review*, 40(1):110–112, 1998.

[143] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[144] S. Ta'asan. One shot methods for optimal control of distributed parameter systems 1: Finite dimensional control. 1991.

[145] S. Ta'asan, G. Kuruvila, and M. Salas. Aerodynamic design and optimization in one shot. In *30th aerospace sciences meeting and exhibit*, page 25, 1992.

[146] T. W. Taylor, F. Palacios, K. Duraisamy, and J. J. Alonso. A hybrid adjoint approach applied to turbulent flow simulations. In *21st AIAA Computational Fluid Dynamics Conference*, page 2452, 2013.

[147] H. Tennekes, J. L. Lumley, J. Lumley, et al. *A first course in turbulence*. MIT press, 1972.

[148] R. Thompson, L. Sampaio, W. Edeling, A. Mishra, and G. Iaccarino. A strategy for the eigenvector perturbations of the Reynolds stress tensor in the context of uncertainty quantification. In *Proceedings of the Summer Program*, page 425, 2016.

[149] R. L. Thompson, L. E. B. Sampaio, F. A. de Bragança Alves, L. Thais, and G. Mompean. A methodology to evaluate statistical errors in DNS data of plane channel flows. *Computers & Fluids*, 130:1–7, 2016.

[150] V. Toropov. Simulation approach to structural optimization. *Structural Optimization*, 1(1):37–46, 1989.

[151] B. Tracey, K. Duraisamy, and J. Alonso. Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 259, 2013.

[152] B. D. Tracey, K. Duraisamy, and J. J. Alonso. A machine learning strategy to assist turbulence model development. In *53rd AIAA Aerospace Sciences Meeting*, page 1287, 2015.

[153] M. S. Uberoi. Effect of wind-tunnel contraction on free-stream turbulence. *Journal of the Aeronautical Sciences*, 23(8):754–764, 1956.

[154] S. Wallin and A. V. Johansson. An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *Journal of Fluid Mechanics*, 403:89–132, 2000.

[155] C. Wang, Y. Jang, and M. Leschziner. Modelling two-and three-dimensional separation from curved surfaces with anisotropy-resolving turbulence closures. *International Journal of Heat and Fluid Flow*, 25(3):499–512, 2004.

[156] J.-X. Wang, R. Sun, and H. Xiao. Quantification of uncertainties in turbulence modeling: A comparison of physics-based and random matrix theoretic approaches. *International Journal of Heat and Fluid Flow*, 62:577–592, 2016.

[157] J.-X. Wang, J.-L. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids*, 2(3):034603, 2017.

[158] J.-X. Wang, J. Wu, J. Ling, G. Iaccarino, and H. Xiao. A comprehensive physics-informed machine learning framework for predictive turbulence modeling. *arXiv preprint arXiv:1701.07102*, 2017.

[159] J. Weatheritt and R. Sandberg. The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, 68:298–318, 2017.

[160] J. Weatheritt and R. Sandberg. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. *Journal of Computational Physics*, 325:22–37, 2016.

[161] B. Wieneke and A. Sciacchitano. PIV uncertainty propagation. In *11th Int Symp on PIV-PIV15*, 2015.

[162] D. C. Wilcox. Reassessment of the scale-determining equation for advanced turbulence models. *AIAA journal*, 26(11):1299–1310, 1988.

[163] D. C. Wilcox et al. *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA, 1998.

[164] M. Woopen, G. May, and J. Schütz. Adjoint-based error estimation and mesh adaptation for hybridized discontinuous galerkin methods. *International journal for numerical methods in fluids*, 76(11):811–834, 2014.

[165] J.-L. Wu, H. Xiao, and E. Paterson. Data-driven augmentation of turbulence models with physics-informed machine learning. *arXiv preprint arXiv:1801.02762*, 2018.

[166] J. Wu, R. Sun, S. Laizet, and H. Xiao. Representation of stress tensor perturbations with application in machine-learning-assisted turbulence modeling. *Computer Methods in Applied Mechanics and Engineering*, 2018.

[167] J. Wu, H. Xiao, R. Sun, and Q. Wang. RANS equations with Reynolds stress closure can be ill-conditioned. *arXiv preprint arXiv:1803.05581*, 2018.

[168] J. Wu, J. Wang, H. Xiao, and J. Ling. Physics-informed machine learning for predictive turbulence modeling: A priori assessment of prediction confidence, 2016.

[169] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, and C. Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed bayesian approach. *Journal of Computational Physics*, 324:115–136, 2016.

[170] Z. Zhang, K. Duraisamy, and N. A. Gumerov. Efficient multiscale Gaussian process regression using hierarchical clustering. *arXiv preprint arXiv:1511.02258*, 2015.

[171] Z. J. Zhang and K. Duraisamy. Machine learning methods for data-driven turbulence modeling. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2460, 2015.

[172] A. Zymaris, D. Papadimitriou, K. Giannakoglou, and C. Othmer. Continuous adjoint approach to the Spalart–Allmaras turbulence model for incompressible flows. *Computers & Fluids*, 38(8):1528–1538, 2009.

[173] A. Zymaris, D. Papadimitriou, K. C. Giannakoglou, and C. Othmer. Adjoint wall functions: A new concept for use in aerodynamic shape optimization. *Journal of Computational Physics*, 229(13):5228–5245, 2010.