Data-Driven Turbulence Modeling

Discovering Turbulence Models using Sparse Symbolic Regression

Elske van Leeuwen

Delft University of Technology

TU Delft

Data-Driven Turbulence Modeling

Discovering Turbulence Models using Sparse Symbolic Regression

by

Elske van Leeuwen

Supervisor Sioux:	Dr. ir. Werner Lazeroms
Supervisor TU Delft:	Dr. Alexander Heinlein
Exam Committee:	Prof.dr.ir. Martin van Gijzen
	Dr.ir. Yoeri Dijkstra
Institution:	Delft University of Technology
Place:	Faculty of Applied Mathematics, Delft
Project Duration:	November, 2021 - December, 2022

Cover Image: Wake Vortex Study at Wallops Island by NASA/Langley Research Center (NASA-LaRC)

TU Delft

Preface

This thesis is the final deliverable to finish the master Applied Mathematics at the Delft University of Technology. The project is conducted in cooperation with Sioux Technologies in Eindhoven, which provides high tech solutions for a wide variety of industrial problems. To provide even better solutions for problems related to computational fluid dynamics, the application of machine learning in turbulence modeling has been explored in this project. A deeper knowledge of data-driven turbulence models is obtained, which can serve as a starting point for eventually applying these models in an industrial setting.

I want to thank my supervisor at Sioux, Werner Lazeroms, for providing the project and for sharing his wide knowledge of turbulence with me. Furthermore, I want to thank my supervisor at the TU Delft, Alexander Heinlein, for the nice discussions and his willingness to always help.

Finally, I want to thank my family, who have always been supportive during my seven plus years of studying at the TU Delft. And of course, all my friends in Delft, for the unforgettable time as a student. Special thanks to David, who has been my rock in this final year.

Elske van Leeuwen
Delft, December 2022

Abstract

Computational Fluid Dynamics (CFD) is the main tool to use in industry and engineering problems including turbulent flows. Turbulence modeling relies on solving the Navier-Stokes equations. Solving these equations directly takes a lot of time and computational power. More affordable methods solve the Reynolds Averaged Navier-Stokes (RANS) equations. The most commonly used RANS models rely on Linear Eddy Viscosity Models for the Reynolds stress closure. However, this type of modeling does not provide satisfactory accuracy in general problems, which include curvature, impingement and separation. The rapid developments of Machine Learning (ML) in fluid dynamics and the increase of available high fidelity data of turbulent flows led to the introduction of data-driven RANS turbulence modeling. ML-augmented RANS models are promising but often lack generalisability or does not meet the preference of being interpretable. In this thesis, a sparse symbolic regression method has been used to generate interpretable algebraic equations for the non linear anisotropic Reynolds stress. These equations are built from a library of candidate functions of which the best fitting functions are selected by solving a sparse regression problem. A deeper understanding of this methodology is achieved by creating models using different input features, including a wall damping function to improve near-wall behaviour of the model. Furthermore, different sparse regression problems are investigated, including constrained regression problems for which physical knowledge is used. The discovered models are propagated in a CFD solver to obtain a corrected velocity field. Most of the discovered models improve the prediction of the Reynolds stress and its anisotropy compared to the $k-\omega$ RANS model. A small number of the discovered models are able to improve the prediction of the streamwise velocity compared to the RANS model. By analysing the discovered models, it is recognised that a specific term in the algebraic model is the basis of an inadequate prediction of the velocity. Furthermore, it is observed that a considerable amount of the implemented models provide an unconverged solution, which is visible as instabilities in the flow variables. Suggestions are given to improve the stability of the models. The discovered models, which include more physical knowledge by applying physical constraints to the regression problem or by including near-wall treatments, give promising results.

Contents

1 Introduction

1.1. Background

Turbulent flows are ubiquitous in nature and technology, and simulating them is an active research topic. Examples of turbulent flows in technology are the flows in nozzles and pipes, flows in devices such as turbomachinery and heat exchangers, and flows around moving objects, such as vehicles and airplanes. Accurate predictions of the turbulent flows are important for the design of that perfect car or airplane having as less drag as possible. Also in the design of turbo machines or full wind farms, insight in the turbulent flows is important [1]. Computational Fluid Dynamics (CFD) is an important approach in obtaining this insight of the physical characteristics of turbulence [2]. It relies on governing equations and mathematical models that capture the complexity of turbulent flows, which are solved numerically. CFD is favoured compared to experiments as alternative analysis tool. The practical advantage of CFD over experiments is that it is cheaper, generates more data and can include data that is not measurable. Furthermore, CFD has the ability to modify the virtual environment to investigate physical behaviour that is otherwise impossible to gauge. For instance, consider a safety analysis of a turbo engine or nuclear reactor to investigate the behaviour under severe conditions. Without damaging the environment, this can not be tested experimentally. Hence, CFD is and stays an important analysing and modeling technique.

Modeling turbulent flow starts with describing the motion of the flow, which is done by the Navier-Stokes equations. Solving these equations directly can be done with the Direct Numerical Simulation (DNS). However, in many cases, this is excessively expensive because of the large range of turbulent scales [3]. It can take several months to perform one DNS simulation of a simple flow. Therefore, turbulence models are used to perform computationally affordable simulations. These models aim to accurately describe the effect of the chaotic behaviour of turbulence on the mean flow. Two strategies for turbulence modeling exist, namely Large Eddy Simulations (LES) and Reynolds Averaged Navier-Stokes models (RANS). In LES the large scales of a turbulent flow are solved while influence of the smallest scales are incorporated through a model. Although LES has been shown to be a powerful and successful method in simulating a variety of complex turbulent flows, its application is limited since it still requires a high grid resolution or small time steps and thus large computational cost. RANS models on the other hand are currently the most widely used models in industry and are expected to remain standard for simulating turbulent flows, because of their lower computational cost. However, in some cases they suffer from poor accuracy and predictive power.

RANS models are based on the RANS equations, which are obtained by decomposing the flow quantities into their time-averaged and fluctuating components and averaging the Navier-Stokes equations. This yields a nonlinear Reynolds stress term that requires additional modeling to fully resolve the system. Such models, referred to as turbulent closure models, generally involve a relation that connects the Reynolds stresses to the mean-flow field with one or two additional equations. The most common used RANS models rely on the Linear Eddy Viscosity Model (LEVM) for the Reynolds stress closure, which assumes a linear relationship between the Reynolds stresses and a mean-velocity gradient ten-

sor [4]. Although this type of model works well on simple flows, it does not provide satisfactory accuracy in more general configurations, such as those with curvature, impingement and separation. In those cases, more advanced nonlinear modeling of the Reynolds stress is required, such as with the explicit algebraic Reynolds stress model of Wallin and Johansson [5], or with the differential Reynolds stress models [6]. However, these nonlinear models are not used often for industrial problems, since they do not give consistent improvements over the LEVM and often have convergence problems [7]. Furthermore, many undetermined parameters need to be tuned based on datasets from particular classes of problems. This limits the usage of CFD in industry, which desires an affordable turbulence model applicable for a wide range of geometries [8].

There has been an increased interest in using Machine Learning (ML) to improve, augment or develop RANS turbulence models. The computational power has been majorly improved over the past decades, which contributed to an increase of available high-fidelity data from DNS and fast developments in ML techniques. This led to the introduction of data-driven RANS turbulence modeling [9]. With high-fidelity data, correction terms can be trained to compensate for the error introduced by the used model, which results in a data-augmented RANS model. Previous research showed that ML-augmented turbulence modeling is promising and can offer improved predictions over classical models [10].

Many challenges remain in the current state-of-the-art of data-driven RANS models, to begin with the generalisability of data-driven models [11]. Ideally, an established data-driven approach can be applied to different or unseen scenarios. However, previous work demonstrated that the predictive ability of the model is limited to flow cases that are similar to the training flow. Ling *et al.* for instance, showed that their ML model works adequately for similar flows, but fails for the cases which deviate significantly from the training flows [12]. Furthermore, previous research revealed that an interpretable model is preferred and can enhance the generalisation capabilities. Another challenge is to create a robust model. When a data-driven RANS model is obtained, it often has to be implemented in a CFD solver to recover the velocity profiles. However, Wu *et al.* showed that small errors in the Reynolds stress can lead to significant errors in the velocity because of ill-conditioning of the RANS equations with its closure model [13].

Recently, the relatively new sparse symbolic regression technique [14], has been introduced in data-driven turbulence modeling and the initial results are promising [15] [16]. Sparse symbolic regression generates an algebraic equation to describe the quantity of interest. This equation is interpretable, which makes the method favourable compared to other machine learning techniques, such as Neural Networks (NN), which are black-box models and hard to interpret. Furthermore, the derived algebraic equation is lighter and more efficient to integrate in an existing CFD solver, which can make it easier to assess the robustness of the model. Finally, it has been shown that integration of physical knowledge is possible, which can contribute to more general models. These early successes motivate to continue exploring the possibilities of sparse symbolic regression in turbulence modeling.

In this research, sparse symbolic regression is applied to find an algebraic closure model for the RANS equations. This will be tested on different geometries of which DNS data is available. We aspire to obtain a deeper understanding of the methodology and the obtained models. This will be achieved by creating models using different input features, including a wall damping function. On numerical level, this will be achieved by analysing the performance of different regression techniques, which can be used in sparse symbolic regression. Furthermore, constraints will be added to regression techniques such that the model is constricted to prior physical knowledge of the Reynolds stress. Finally, we would like to get more insight in the performance of the models in terms of generalisability and robustness.

1.2. Research Objectives

The main objective of this research project is to develop data-driven turbulence models using sparse symbolic regression to improve the $k-\omega$ RANS turbulence model, which contains physical knowledge of the Reynolds stress and should be generalisable and robust. To achieve this, several sub-goals have been set up:

- Create a modeling framework to learn algebraic models for the anisotropic Reynolds stress using sparse symbolic regression, while embedding physical knowledge. The framework should consist of a model discovery part and a model assessment part.
- Generate different test cases to train and test the models.
- Implement the models in the CFD solver OpenFOAM and assess the improvements in mean-flow over the standard RANS model.
- Assess the generalisability and robustness of the models.

Relevant questions and sub-questions based on the objectives are:

Q1. Which features are the most relevant when generating an algebraic equation for the anisotropic Reynolds stress?

  - Is the sparse symbolic regression technique able to select the most relevant features and how do we verify that those features are the most relevant?
  - If not, how can we do a pre-selection of the features?
  - Can we add features to include near-wall treatment?

Q2. Which sparse symbolic regression technique finds the best performing algebraic model?

  - Which sparse symbolic regression techniques exist and how does each regression function and optimiser influence the model?
  - Can we apply constraints to embed physical knowledge and how does such constraints influence the performance of the model?

Q3. How does the resulting turbulence model perform in terms of robustness and generalisability?

  - Can we a priori predict the performance of the regressed model in terms of robustness and generalisability?

1.3. Structure

This thesis starts with providing the necessary background information of turbulence and its modeling in Chapter 2. An introduction of machine learning is given in Chapter 3. In this chapter, also a state-of-the-art of machine learning in turbulence modeling is given. Chapter 4 covers the methodology of the research. A description is given of sparse symbolic regression as applied in this thesis, including an explanation of the used regression problems and how physical constraints and near-wall treatment are incorporated. Furthermore, the implementation of the models in a CFD solver is presented. The cases on which the methodology is applied are described in Chapter 5. Chapter 6 presents the results as obtained in this research. Finally, the drawn conclusions are given in Chapter 7, together with recommendations for future research.

2

Turbulence and Turbulence Modeling

This chapter serves as an introduction to turbulence and the two common used turbulence modeling strategies, namely RANS and LES. More attention is given to RANS models, since these models will be the basis of our modeling approach. The governing equations will be explained, including the closure problem.

2.1. Turbulent Flows

Turbulence is a state of fluid motion which is characterised by a chaotic behaviour of the flow velocity and the pressure of the flow. This is the opposite of a so called laminar flow, see figure 2.1. In the laminar case, the fluid flows smoothly in parallel layers with no or little mixing. The motion of a fluid can be described by a complete set of equations, called the Navier-Stokes (NS) equations. The NS equations are a system of nonlinear partial differential equations and describe the relation between flow variables, such as the velocity and pressure, as a function of position and time [1]. The NS equations emanate from the laws of conservation of mass, momentum and energy. When mass is conserved, it means that the total mass of a volume element in the flow remains unchanged over time. This can be expressed as

$$\frac{d\rho}{dt} + \nabla \cdot (\rho \boldsymbol{U}) = 0, \tag{2.1}$$

where ρ, t and $\boldsymbol{U}$ are the density, time and velocity respectively. The term $\nabla \cdot (\rho \boldsymbol{U})$ represents the overall rate of mass additions per volume element due to convection [2]. When the divergence of the velocity is zero, the fluid is called incompressible. In that case, the conservation of mass can be reduced to the continuity equation

$$\nabla \cdot \boldsymbol{U} = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} = 0, \tag{2.2}$$

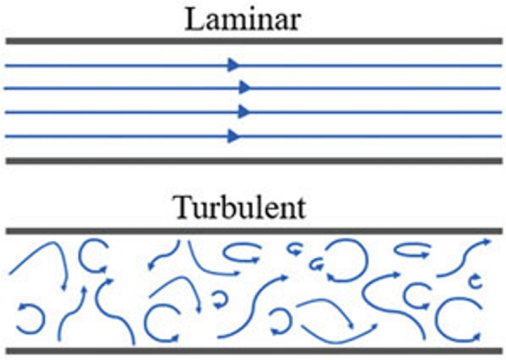


Figure 2.1: Schematic drawing of a laminar and turbulent flow, taken from [2, p. 35].

where U, V and W are the velocity components in the x, y and z direction, respectively. In this report, we limit the scope to incompressible flows.

The conservation of momentum is nothing more than Newton's second law of motion, $F = m\boldsymbol{a}$, applied to fluid elements. That is, the total force F acting on a fluid equals the change of momentum defined as the product of the mass m and the acceleration $\boldsymbol{a}$ of the fluid. For an incompressible flow, the conservation of momentum can be described as

$$\rho\frac{D\boldsymbol{U}}{Dt} = \rho(\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U}\cdot\nabla)\boldsymbol{U}) = \rho g - \nabla p + \mu\nabla^2\boldsymbol{U}. \tag{2.3}$$

Equations 2.2 and 2.3 are known as the Navier-Stokes equations, where μ is defined as the dynamic viscosity, which is a material property [1]. When no free surface is present in the flow, equation 2.3 can be further simplified to

$$\rho\frac{D\boldsymbol{U}}{Dt} = \rho(\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U}\cdot\nabla)\boldsymbol{U}) = -\nabla p + \mu\nabla^2\boldsymbol{U}. \tag{2.4}$$

In that case, the gravity term is absorbed in the pressure term. The NS equations 2.4 and the continuity equation 2.2 are the basis for describing the motion of an incompressible flow of a homogeneous fluid.

In the NS equation, the term $\mu\nabla^2\boldsymbol{U}$ represents a force of friction and acts as a damping force, driven by viscosity [2]. Thus, the higher the viscosity, the higher the damping. The term $\rho(\boldsymbol{U}\cdot\nabla)\boldsymbol{U}$ represents the inertial force, the force due to the momentum of the fluid. The denser a fluid and the higher its velocity, the more momentum the fluid has. The ratio between these two term is expressed by the Reynolds number, defined as

$$Re = \frac{\rho U L}{\mu}, \tag{2.5}$$

where U and L are the characteristic velocity and dimension of the flow. When the viscous force is much smaller than the inertial force, a flow becomes turbulent, indicated with a high Reynolds number.

Turbulence is often described in terms of 'eddying' motions. When a flow becomes unstable due to a high Reynolds number, energetic local swirls of the flow arise, which are the turbulent eddies. It starts with large eddies, which transform into smaller and smaller eddies. Therefore, turbulent flows consist of the superposition of a continuous scale of small to large eddies. The scales of the smallest turbulent motions are the Kolmogorov scales.

Turbulent flows involve a very wide range of active spatial and temporal scales. In combination with the chaotic behaviour of a turbulent flow, solving the NS equation is very complex. It is possible to solve the NS equation directly with Direct Numerical Simulation (DNS). However, resolving all the scales of motion, including the smallest ones, requires a high resolution and a lot of computational time, despite the growth of computer power in the last decades. Furthermore, it is estimated that the number of floating-point operations grows as Re^3 [17]. Therefore, to deal with the complexity of the problem, several theoretical and computational approaches have been introduced to characterise turbulence. Among these simplified approximations, Reynolds Averaged Navier-Stokes (RANS) models and Large Eddy simulation (LES) are most common [18]. In the following sections, RANS and LES are discussed in more detail.

2.2. RANS

RANS models are based on the RANS equations. The RANS equations are obtained by splitting the instantaneous flow U into a mean quantity $\overline{U}$ and its associated fluctuations U' illustrated in Figure 2.2. This is called Reynolds decomposition, named after its originator Osborne Reynolds and can be written as

$$\boldsymbol{U} = \overline{\boldsymbol{U}} + \boldsymbol{U}'. \tag{2.6}$$

The time-averaged velocity can be simply obtained by

$$\boldsymbol{U}(\boldsymbol{x}) = \lim_{T\to\infty}\frac{1}{T}\int_t^{t+T}\boldsymbol{U}(\boldsymbol{x},t)dt. \tag{2.7}$$

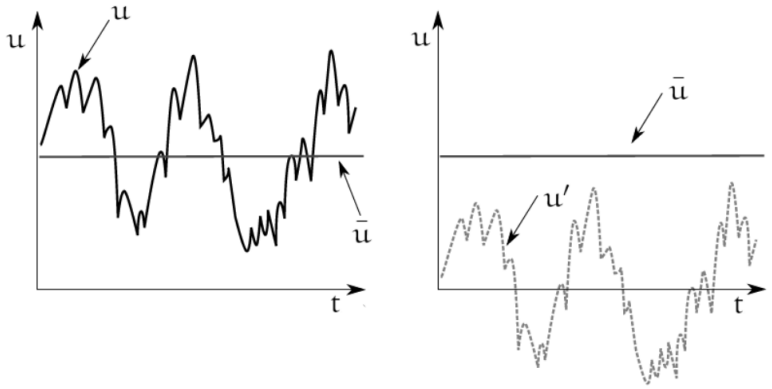


Figure 2.2: Process of Reynolds averaging in which the velocity u is decomposed into a its mean $\overline{u}$ and fluctuating u' part; graphic taken from [19, p. 42].

Substituting the decomposition into the NS-equations and continuity equation and performing an additional time-averaging operation, results in the RANS equations. In tensor notation, the RANS equations read

$$\frac{D\overline{U_i}}{Dt} = -\frac{1}{\rho}\frac{\overline{p}}{x_j} + \frac{\partial}{\partial x_j}(\nu\frac{\partial\overline{U_i}}{\partial x_j} - \overline{U_i'U_j'}), \tag{2.8}$$

$$\frac{\partial\overline{U_i}}{\partial x_i} = 0. \tag{2.9}$$

Note that the RANS equations include a new term $R_{ij} = \overline{U_i'U_j'}$, called the Reynolds stress. The Reynolds stress cause that we have six additional unknowns, which makes it impossible to solve the equations. This is referred to as the turbulence closure problem. Hence, additional models for the Reynolds stress are required.

2.2.1. Linear Eddy Viscosity Models

A majority of the turbulence models, which models the Reynolds stress in the RANS equations, are the Linear Eddy Viscosity Models (LEVM). These models rely on a linear constitutive relationship, also known as the Boussinesq hypothesis, yielding

$$R_{ij} = \overline{U_i'U_j'} = 2\nu_t S_{ij} - \frac{2}{3}k\delta_{ij}. \tag{2.10}$$

Here, δ_{ij} is the Kronecker delta, S_{ij} is the instantaneous strain rate tensor defined by

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

k is the mean turbulent kinetic energy

$$k = \frac{1}{2}(\overline{U_i'U_i'}), \tag{2.12}$$

and ν_t is called turbulent viscosity or eddy viscosity. With the Boussinesq approximation, the unknown Reynolds stress components are resolved, but again a new variable, the eddy viscosity ν_t, is introduced. Over time, a wide variety of possible expression for the eddy viscosity has been derived. These eddy viscosity models can be classified in the amount of additional used equations. Zero-equation models or algebraic models require no additional equations and are calculated directly from the flow variables. Consequently, these models are not able to account for history effects of the turbulence. An example of such a model is the mixing-length model for shear flows given by the equation

$$\nu_t = \left|\frac{\partial\overline{U}}{\partial y}\right| l_m^2, \tag{2.13}$$

where l_m is the mixing length and $\frac{\partial \overline{U}}{\partial y}$ the partial derivative of the streamwise velocity with respect to the wall normal direction [20]. The mixing length l_m has to be specified, which is inevitably dependent on the geometry of the flow. For complex flows this specifications requires a large measure of guesswork and therefore, their application is limited to very simple flow geometries, such as simple wall-bounded flows [21].

In one-equation models, typically a transport equation for the turbulence kinetic energy k or eddy viscosity is solved. An example of such a model is Prandtl's one-equation model [22] that solves the eddy viscosity by

$$\nu_t = l_m \sqrt{k}, \tag{2.14}$$

where k is described by the partial differential equation

$$\frac{\partial k}{\partial t} + \overline{U_j} \frac{\partial k}{\partial x_j} = R_{ij} \frac{\partial \overline{U_i}}{\partial x_j} - \epsilon + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right]. \tag{2.15}$$

Prandtl's equation provides closure by defining the dissipation ϵ, the turbulent kinematic viscosity, the eddy length scale and additional closure coefficients.

Two-equation models include two additional transport equations. Usually a transport equation for the kinematic energy is chosen as well as one additional transport variable, which could be the turbulence dissipation ϵ or the turbulence dissipation rate ω. One of the most widely used two-equation transport model is the $k - \omega$ model developed by Wilcox and others [20], which solves the transport equations of the eddy frequency ω and the turbulent kinetic energy k, defined as

$$\frac{\partial k}{\partial t} + \overline{U_j} \frac{\partial k}{\partial x_i} = R_{ij} \frac{\partial \overline{U_i}}{\partial x_j} - \beta^* k\omega + \frac{\partial}{\partial x_j} \left[(\nu + \nu_t \sigma^*) \frac{\partial k}{\partial x_i} \right], \tag{2.16}$$

$$\frac{\partial \omega}{\partial t} + \overline{U_i} \frac{\partial \omega}{\partial x_i} = \gamma \frac{\omega}{k} R_{ij} \frac{\partial \overline{U_i}}{\partial x_j} - \beta \omega^2 + \frac{\partial}{\partial x_i} \left[(\nu + \nu_t \sigma) \frac{\partial \omega}{\partial x_i} \right]. \tag{2.17}$$

These equations contain the five closure coefficients β, β^*, γ, σ and σ^*. The closure relationship for this model is

$$\nu_t = \frac{k}{\omega}. \tag{2.18}$$

Another widespread two-equation model is the $k - \epsilon$ model, in which transport equations for the turbulent kinetic energy and the dissipation are solved, defined as

$$\frac{\partial k}{\partial t} + \overline{U_j} \frac{\partial k}{\partial x_j} = R_{ij} \frac{\partial \overline{U_i}}{\partial x_j} - \epsilon + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right], \tag{2.19}$$

$$\frac{\partial \epsilon}{\partial t} + \overline{U_j} \frac{\partial \epsilon}{\partial x_j} = C_1 \frac{\epsilon}{k} R_{ij} \frac{\partial \overline{U_i}}{\partial x_j} - C_2 \frac{\epsilon^2}{k} + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right], \tag{2.20}$$

where C_1, C_2, σ_k and σ_ϵ are constants. The eddy viscosity is calculated with

$$\nu_t = C_\mu \frac{k^2}{\epsilon}. \tag{2.21}$$

Overtime, many more RANS models have been developed, which rely on the Boussinesq hypothesis as defined in equation 2.10. Unfortunately, the intrinsic assumption of the Boussinesq hypothesis that the anisotropy is locally determined by $\partial \overline{U_i} / \partial x_j$ is not valid for all cases. Therefore the eddy viscosity models yield poor accuracy for many flows. Only for simple shear flows, the hypothesis is reasonable.

2.3. LES

Although the focus in this project is on RANS models, a brief explanation of LES models is given for completeness. The larger eddies in turbulent flows obtain their kinetic energy from the bulk fluid energy, which contains most of the turbulent kinetic energy of the flow. By breaking up the large eddies into

smaller eddies, energy is transferred to the smaller eddies [2]. So, the larger eddies are responsible for the majority of the diffusive processes. The smaller eddies on the other hand, take the the kinetic energy from the larger eddies and dissipate it into heat at the smallest scales. The smaller eddies are statistically isotropic and therefore, have a more universal character and are more independent of the boundary conditions and the mean flow velocity than the larger eddies. Based on this theory, LES models have been developed to resolve the large eddies and approximate the behaviour of the small eddies. This is done by separating the velocity field into a field of resolved large eddies and a sub-grid part representing the small scale motions, which are modeled, resulting in

$$\boldsymbol{U} = \overline{\boldsymbol{U}} + \boldsymbol{U}'. \tag{2.22}$$

The decomposition of the velocity is obtained by employing a filtering operation

$$\overline{\boldsymbol{U}}(\boldsymbol{x}, t) \equiv \int \boldsymbol{U}(\boldsymbol{x}', t) G(\boldsymbol{x} - \boldsymbol{x}') d\boldsymbol{x}', \tag{2.23}$$

where $G(\boldsymbol{x} - \boldsymbol{x}')$ is a filter function. The equations of motion for the resolved field are derived by substituting the decomposition into the NS equations, and subsequently filtering the resulting equation. LES is able to capture a significant number of velocity fluctuations and therefore provides more accurate simulations than RANS models do. However, LES is in general more time intensive than RANS models, since it requires a finer grid scale. Compared with DNS, the vast computational cost to represent the small-scale motions is avoided and hence, LES is less time intensive than DNS.

2.4. Anisotropy Tensor

The Reynolds stress, $R_{ij} = \overline{U_i' U_j'}$ can be divided into an isotropic and anisotropic part

$$R_{ij} = \frac{2}{3} k \delta_{ij} + a_{ij}. \tag{2.24}$$

Here, $a_{ij} = 2k b_{ij}$ is the Reynolds stress anisotropy tensor and $k = \frac{1}{2}\text{trace}(R_{ij})$ is the turbulent kinetic energy [21]. The anisotropic part of the Reynolds stress is the only part that is active in transporting momentum and hence, is the most important part. Equation 2.24 can be rewritten to define the non-dimensional anisotropic Reynolds stress tensor, b_{ij}, as

$$b_{ij} = \frac{R_{ij}}{2k} - \frac{1}{3}\delta_{ij}. \tag{2.25}$$

As mentioned previously, eddy-viscosity models typically rely on the assumption that b_{ij} is a function of local mean-flow quantities. LEVM, such as the $k-\omega$ and $k-\epsilon$ models, use the Boussinesq assumption that $b_{ij} = \frac{\nu_t}{k} S_{ij}$. However, this assumption introduces modelling errors. In this research, we aim to reduce the error by finding a (nonlinear) model for the anisotropy Reynolds stress using DNS databases and machine learning.

2.4.1. Realisability

From the properties of the Reynolds stress R_{ij} and its anisotropic part b_{ij}, physical constraints can be derived [23]. This starts with the definition that a square matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is positive semi-definite if and only if $\boldsymbol{x^T A x} \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^N$. The Reynolds stress is constructed by taking the outer product of turbulent fluctuation $\boldsymbol{U'}$ with itself and therefore satisfies positive semi-definiteness, since

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

It is still positive semi-definite after taking the temporal average $R_{ij} = \overline{U_i' U_j'} = \frac{1}{n}\sum_n U_i' U_j'$, since all the time-samples adhere this property. Since the Reynolds stress tensor is positive semi-definite, all its eigenvalues are non-negative and therefore also its determinant and trace. Furthermore, the Reynolds stress will have non-negative diagonal components and the Cauchy-Schwarz inequality must hold. In summary:

$$R_{\alpha\alpha} \geq 0 \quad \forall \alpha \in \{1,2,3\}, \qquad \det(R) \geq 0, \qquad R_{\alpha\beta}^2 \leq R_{\alpha\alpha} R_{\beta\beta} \quad \forall \alpha \neq \beta. \tag{2.26}$$

Table 2.1: Special states of the Reynolds-stress tensor in terms of the eigenvalues of anisotropy tensor $\boldsymbol{b}$. With one- and two-component turbulence is meant that fluctuations exists along one or two directions respectively. Axisymmetric two-component turbulence indicates that fluctuations exist along two directions with equal magnitude, also referred to as pancake-like turbulence.

State of turbulence	Eigenvalues of $\boldsymbol{b}$	Designation in Figure 2.3
One-component	$\lambda_i = \left[\frac{2}{3}, -\frac{1}{3}, -\frac{1}{3}\right]^T$	Corner x_{1C}
Axisymmetric two-component	$\lambda_i = \left[\frac{1}{6}, \frac{1}{6}, -\frac{1}{3}\right]^T$	Corner x_{2C}
Isotropic	$\lambda_i = 0$	Corner x_{3C}
Axisymmetric expansion	$0 \leq \lambda_1 \leq \frac{1}{3}$ and $-\frac{1}{6} \leq \lambda_2 = \lambda_3 \leq 0$	Black line x_{1C} - x_{3C}
Axisymmetric contraction	$-\frac{1}{3} \leq \lambda_1 \leq 0$ and $0 \leq \lambda_2 = \lambda_3 \leq \frac{1}{6}$	Blue line x_{2C} - x_{3C}
Two-component	$\lambda_1 + \lambda_3 = \frac{1}{3}$ and $\lambda_2 = -\frac{1}{3}$	Red line x_{1C} - x_{2C}

These properties of R_{ij} have implications for the anisotropic stress b_{ij}. When ϕ_i are the eigenvalues of R_{ij}, then the eigenvalues of b_{ij}, λ_i, can be written as

$$\lambda_i = \frac{\phi_i}{2k} - \frac{1}{3}. \tag{2.27}$$

This has as consequence that the eigenvalues and the diagonal components of b_{ij} are in the interval $[-\frac{1}{3}, \frac{2}{3}]$. Furthermore, we know by the Cauchy-Schwarz inequality in equation 2.26, that the off-diagonal components of b_{ij} are in $[-\frac{1}{2}, \frac{1}{2}]$.

To analyse whether the anisotropic stress tensor is in the derived intervals, it is useful to characterise the tensor more simply than by its six components b_{ij}. In fact, the anisotropy can be characterised by just two independent invariants, which are scalars obtained from a tensor (e.g., $b_{ii}.b_{ii}^2, bii^3$) and its values are the same in any coordinate system [21]. These invariants are defined by

$$II = b_{ij}b_{ji}/2, \qquad III = b_{ij}b_{in}b_{jn}/3, \tag{2.28}$$

and allow for a simple graphical representation. At any point in a turbulent flow, the invariants can be determined from the Reynolds stress and plotted as a point in the $III - II$ plane as shown in Figure 2.3. This representation was introduced by Lumley and Newman and therefore is known as the Lumley triangle [24] [25]. The invariants fall within a triangular domain, corresponding to the constraints mentioned earlier. Rewriting the intervals of the anisotropy tensor, the following nonlinear relationships can be constructed, which define the triangle domain

$$II \geq \frac{3}{2}\left(\frac{4}{3}|III|\right)^{2/3}, \qquad II \leq \frac{2}{9} + 2III. \tag{2.29}$$

Points outside the triangle correspond to non-realisable Reynolds stresses, which could have negative or complex eigenvalues. The corners and sides of the triangle correspond to special states of the Reynolds-stress tensor. These different states in terms of invariants and eigenvalues of the anisotropy tensor b_{ij} are listed in Table 2.1.

2.5. Near-Wall Treatment

The presence of a wall causes a number of different effects in a turbulent flow, which makes it complex to model. The flow can be characterised into various regions proceeding perpendicularly from the wall. The region adjacent to the wall is called the viscous sublayer, followed by the buffer layer and the log layer. In every layer, different reciprocal actions take place between the viscous stress $\rho\nu(d\overline{U}/dy)$ and the Reynolds shear stress $-\rho\overline{U'V'}$, which together define the total shear stress $\tau(y)$ [2]. At the wall, the no-slip boundary condition dictates that all the Reynolds stresses are zero. Therefore, the wall shear stress entirely consists of the viscous stress

$$\tau_w \equiv \rho\nu\left(\frac{d\overline{U}}{dy}\right)_{y=0}. \tag{2.30}$$

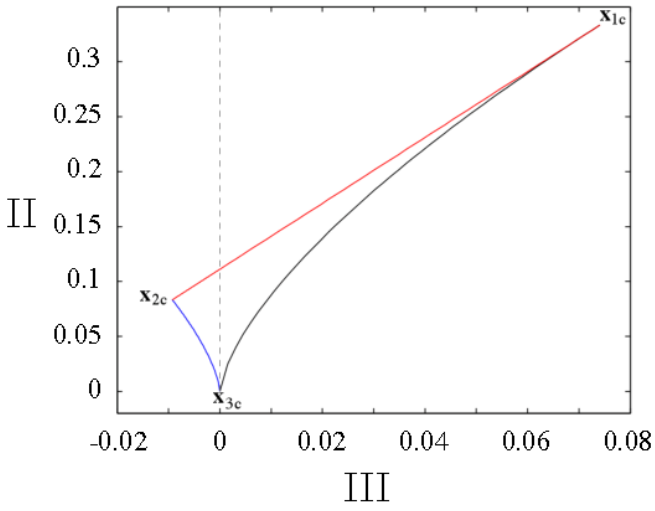


Figure 2.3: Anisotropy-invariant map based on (III, II). The corners and sides of the triangle correspond to special states of the Reynolds-stress tensor as explained in Table 2.1; taken from [26, p. 125].

Away from the wall, the viscous stress is negligible and the total shear stress is therefore described by the Reynolds stress. The different regions are defined on the basis of the distance from the wall measured in wall units, which is denoted by

$$y^+ \equiv \frac{y}{\delta_\nu} = \frac{U_\tau y}{\nu}. \tag{2.31}$$

Here, y is the distance from the wall, U_τ is the friction velocity and δ_ν the viscous lengthscale, which are defined as

$$U_\tau \equiv \sqrt{\frac{\tau_w}{\rho}}, \qquad \delta_\nu \equiv \nu\sqrt{\frac{\rho}{\tau_w}} = \frac{\nu}{U_\tau}. \tag{2.32}$$

The thickness of the layers is dependent of the Reynolds numbers which is illustrated in Figure 2.4. Furthermore, with equation 2.31, a dimensionless velocity can be acquired resulting in

$$U^+ \equiv \frac{\overline{U}}{U_\tau}. \tag{2.33}$$

Due to the different flow physics in each layer, U^+ depends differently on y^+ in each layer. In the viscous layer, for which $y^+ < 5$, the velocity adheres to a linear relation

$$U^+ = y^+. \tag{2.34}$$

In the log layer, for which $y^+ > 30$, the velocity distribution obeys the logarithmic law

$$U^+ = \frac{1}{\kappa}\ln y^+ + C, \tag{2.35}$$

where κ is the von Karman constant, $\kappa = 0.41$, and $C = 5.2$.

The different behaviour of the velocity in each layer is correlated to different scaling of the Reynolds stress in each layer [21]. To determine the behaviour of the Reynolds stress departing from the wall we can write the fluctuating velocity components as Taylor series of the form

$$\begin{aligned} U' &= a_1 + b_1 y + c_1 y^2 + \mathcal{O}(y^3), \\ V' &= a_2 + b_2 y + c_2 y^2 + \mathcal{O}(y^3), \\ W' &= a_3 + b_3 y + c_3 y^2 + \mathcal{O}(y^3). \end{aligned} \tag{2.36}$$

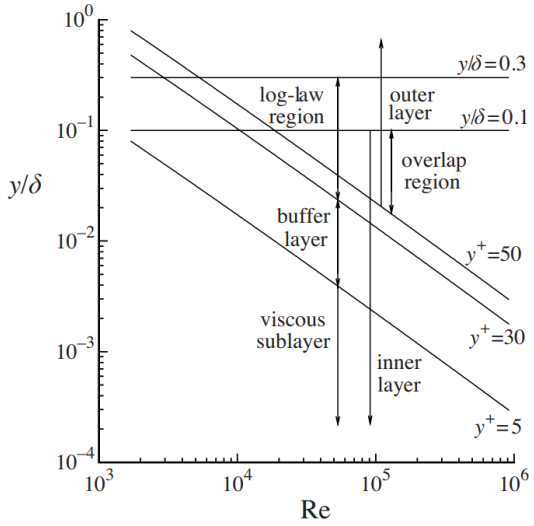


Figure 2.4: Regions and layers in a turbulent channel flow as functions of the Reynolds number, obtained from [21, p. 281]

At the wall, $y = 0$, the no-slip condition yields $U' = a_1 = 0$ and $W' = a_3 = 0$. Similarly, the impermeability condition of the wall yields $V' = a_2 = 0$. Since U' and W' are zero at the wall, its derivatives $(\partial U'/\partial x)_{y=0}$ and $(\partial W'/\partial z)_{y=0}$ are also zero and the continuity equation yields $(\partial V'/\partial y)_{y=0} = b_2 = 0$. As $b_2 = 0$, very close to the wall V' is zero, whereas u' and w' are non-zero, resulting in a two-component flow. Now, the Reynolds stresses near the wall can be obtained from the expansions 2.36 by taking the means of the products of the series and by taking into account the zero valued coefficients (a_1, a_2, a_3 and b_2). This results in

$$\overline{U'U'} = \overline{b_1^2} y^2, \tag{2.37}$$

$$\overline{V'V'} = \overline{c_2^2} y^4, \tag{2.38}$$

$$\overline{W'W'} = \overline{b_3^2} y^2, \tag{2.39}$$

$$\overline{U'V'} = \overline{b_1 c_1} y^3. \tag{2.40}$$

This concludes that $\overline{U'U'}$ and $\overline{W'W'}$ and k behave as y^2, but that $\overline{V'V'}$ and $\overline{U'V'}$ increase more slowly as y^3 and y^4, respectively. Since the basic LEVMs do not incorporate these different scaling for the Reynolds stress components, they require modifications referred to as near-wall treatments.

One way of modifying the basic models to make up for near-wall effect is by the use of wall-functions. The idea of this is to apply boundary conditions some distance away from the wall so that the turbulence model equations are not solved close to the wall. At $y = 0$, the boundary condition for k and ω are

$$\frac{\partial k}{\partial n} = 0, \qquad \frac{\partial \omega}{\partial n} = 0. \tag{2.41}$$

The wall-function boundary conditions are applied at a location $y = y_p$ and are specified as

$$k_p = \frac{u_\tau^2}{\sqrt{C_\mu}}, \qquad \omega_p = \frac{u_\tau}{\sqrt{C_\mu} k y_p} = \frac{\sqrt{k_p}}{C_m^{1/4} k y_p}. \tag{2.42}$$

These values result from asymptotic analysis of the log-layer.

Another way to improve the models behaviour near the wall is by the use of a damping function. The standard $k - \epsilon$ model yields too large values of the turbulent viscosity in the near-wall region [27]. Therefore, Jones and Launder included various damping function and defined the turbulent viscosity as

$$\nu_T = f_\mu C_\mu \frac{k^2}{\epsilon}, \tag{2.43}$$

in which the damping function depends on the turbulence Reynolds number and is specified as

$$f_\mu = \exp\left(\frac{-2.5}{1 + Re_L/50}\right). \tag{2.44}$$

However, different damping functions are introduced. For example, Rodi and Mansour [28] suggested the empirical relation

$$f_\mu = 1 - \exp\left(-0.0002y^+ - 0.00065y^{+2}\right), \tag{2.45}$$

while van Driest [29] was one of the first introducing damping as

$$f_\mu = 1 - \exp\left(\frac{y^+}{A}\right), \tag{2.46}$$

where A is a constant. This function is plotted in Figure 2.5.

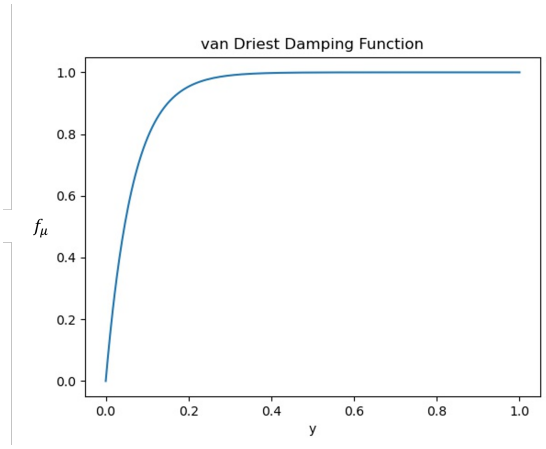


Figure 2.5: Van Driest damping function, equation 2.46

3

Machine Learning in Turbulence Modeling

This chapter provides the necessary background information of Machine Learning (ML) as used in turbulence modeling and describes the current developments of it to put this research into perspective. It also aims to explain current challenges of machine learning and to provide a motivation for the ML method as used in this research.

3.1. Introduction of Machine Learning

Machine Learning is an umbrella term for a wide range of techniques within Artificial Intelligence (AI), which is the theory and development of computer systems to perform tasks that normally require human intelligence, such as speech recognition and decision-making. ML can be generally defined as "the field of study that gives computers the ability to learn without being explicitly programmed" [30]. Its main task is to develop learning algorithms that build models from data. The learning algorithm receives training data to obtain a model that can make predictions on new data [31]. ML algorithms can be roughly categorised into supervised learning and unsupervised learning. Unsupervised learning does not require labeled data and does generally not require any information on the exact solution of the problem. A common area of unsupervised learning is clustering, which is the task of grouping data such that objects in one group are more similar to each other than to the objects in other groups. This technique is often used to find unknown correlations and patterns in datasets. In fluid mechanics clustering can be applied to separate different regions in a flow. Furthermore, unsupervised learning is often used for dimensionality reduction of fluid flow data. In supervised learning the objective is to construct a function, which maps the inputs to given outputs. ML algorithms are trained with available data containing features with associated labels. Common supervised ML methods are linear regression, random forests, support-vector machines and artificial neural networks. These methods are also frequently used in data-driven turbulence modeling and will be described in more details in upcoming sections.

3.1.1. Artificial Neural Networks

Artificial Neural Networks (ANN) are one of the most well-known methods in supervised learning. Neural Networks (NN) consist of simple elements, which are parallel interconnected in a hierarchical, layered organisation [32]. Those simple elements are called neurons and are intended to work similarly as neurons in the biological nervous systems. ANNs are characterised by their flexibility and hence many variations of NNs exist. The classic 'feed forward' NN connects several layers in which each layer is comprised of a number of neurons. The first layer of the network is the input layer, which receives a data vector X. This data vector is passed through the neurons in the hidden layers and eventually reach the output layer. A schematic overview of such NN can be seen in Figure 3.3a. In each neuron of the hidden layers a simple mathematical model is created. The neuron receives the outputs of previous layers as input X_i, which are multiplied by weights ω_i and summed up with a bias b. Subsequently, a

nonlinear activation function is applied on the outcome resulting in the output Y_i. The output of a neuron is eventually calculated by

$$Y = g(\sum_{i=1}^{N} \omega_i X_i + b), \tag{3.1}$$

and is passed as input for the neuron in the next layer. The NN is trained to determine the optimal weights and biases to accurately predict the output of the network. When multiple hidden layers are used, the NN is called a multilayer perceptron (MLP). Deep learning is the branch of ML where many layers are used such that the network can learn more complex relationships.

3.1.2. Decision Trees

A decision tree classifies data by dividing a training set recursively according decision rules. This ML algorithm follows a tree structure to classify data, which makes it a very intuitive ML method. The structure typically consists of one root node, multiple internal nodes and multiple leaf nodes [31]. Each node corresponds to a decision rule, leading to an other decision rule or the final decision outcomes, the leaf nodes; see Figure 3.1 for an example. The ML algorithm aims to produce a tree with decision rules that optimally divides the training data. New unseen data can then be classified according the same decision rules. Those standard decision trees are prone to overfitting and therefore more elaborate variants are proposed, with random forests as one of the most common representatives. Instead of one, an ensemble of decision trees is trained, each on a randomly selected subset of the training data, to improve the generalisability [33]. When the target variables are continuous values, the tree is called a regression tree.

3.1.3. Symbolic Regression

Symbolic regression is a type of regression analysis that tries to find a mathematical expression that fits a given dataset optimally. Two popular methods are Gene Expression Programming and Deterministic Symbolic Regression.

Gene Expression Programming

Gene Expression Programming (GEP) is an Evolutionary Algorithm (EA) that mimics nature's survival of the fittest to end up with an algebraic equation that reproduces the data. GEP is encoded in simple linear structures, the chromosomes, containing one or more genes, which represent mathematical operators and constants [35]. In the training process, these chromosomes are randomly mutated and the best ones are selected. The algorithm returns a mathematical equation and an example of this is shown in Figure 3.2. The figure exhibit an expression tree which represents the equation $f(x, y) = \cos 2 - y(x + 4)$.

Deterministic Symbolic Regression

GEP is an attractive method for turbulence modeling, because of its open-box approach. However, GEP can be expensive, may be prone to overfitting and it generally discovers for each run another model with different mathematical expressions because of its non-deterministic behaviour. A deterministic

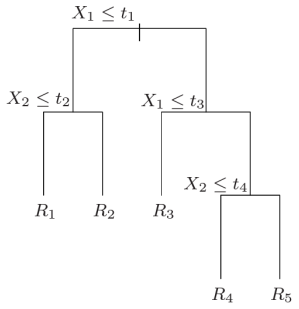


Figure 3.1: Example of a tree structure, obtained from [34, p. 306].

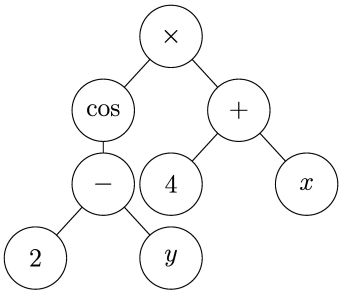


Figure 3.2: Example of Gene Expression Programming, obtained from [36, p. 25].

variant, which does not rely on random processes, is sparse symbolic regression as used in the data-driven modeling technique of Brunton *et al.* [14]. In this research, data is used to discover governing equations of nonlinear, dynamical systems of the form

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t)). \tag{3.2}$$

Brunton *et al.* assume that most physical systems only have a few relevant terms that define the dynamics. To determine the function $\boldsymbol{f}$ from data $\boldsymbol{x}(t)$, a library $\Theta(\boldsymbol{x})$ is constructed. This library consists of candidate functions, which could be polynomials, trigonometric functions or any other nonlinear function dependent on state $\boldsymbol{x}$. An example of a library is

$$\Theta(\boldsymbol{x}) = \begin{bmatrix} 1 & \boldsymbol{x} & \boldsymbol{x}^2 & \dots & \sin \boldsymbol{x} & \cos \boldsymbol{x} \end{bmatrix}. \tag{3.3}$$

Thereafter, a sparse regression problem is set up to determine the sparse vector of coefficients $\Xi = [\xi_2, \xi_2, ..\xi_n]$ that indicates which nonlinear functions are active such that

$$\frac{d}{dt}\boldsymbol{x}(t) = \Theta(\boldsymbol{x})\Xi. \tag{3.4}$$

3.2. Machine Learning in Turbulence Modeling

Simultaneously with the tremendous increase in the amount and availability of data across scientific disciplines, the interest and progress in the field of ML advanced. Also the use of ML in the field of fluid mechanics has grown and especially in turbulence modeling. There are multiple reasons why ML is suitable for modeling turbulence. Brunton argued that "both ML and fluid mechanics tend to rely on the same assumption that there are patterns that can be exploited, even in high-dimensional systems" [37]. Andrea Beck *et al.* pointed out that simulating and measuring turbulence requires large amounts of high dimensional data from which ML can extract low-dimensional information to gain knowledge [11]. The flexible modeling framework of ML can be applied to many different challenges in fluid mechanics, including discovery of governing equation, flow decomposition, producing reduced-order models and flow control. However, in this section, we focus on augmenting RANS turbulence models.

The relevant studies of ML-augmented turbulence modeling, can be categorised based on the level on which the modeling occurs [11] [18]. That is, improving a RANS model can be done by only optimising some parameters θ in the model M or completely replacing variables by modeled terms $\hat{M}$. Furthermore, the RANS model can be completely avoided and the outcome of the governing equation can directly be modeled. A possible hierarchy for the different levels of turbulence modeling can mathematically be written as

$$\begin{aligned} L1 : \theta_{opt} &:= \arg\min_{\theta} ||M - g(\theta)||, \\ L2 : (\theta_{opt}, g_{opt}) &:= \arg\min_{(g(\theta),\theta)} ||M - \hat{M}(\theta, g(\theta))||, \\ L3 : (\theta_{opt}, g_{opt}) &:= \arg\min_{(g(\theta),\theta)} ||U - \hat{U}(\theta, g(\theta))||. \end{aligned} \tag{3.5}$$

The first level of modeling replaces the unclosed Reynolds stresses of equation 2.8 by an a priori determined function g containing parameters θ, in which the parameters are fit through optimisation approaches. This method of modeling, referred to as parameter estimation, is a commonly used approach, since existing turbulence models are often naturally incorporated. Furthermore, the effect of the model on the governing NS equations is typically computed via a modified model for the turbulent viscosity, which is often relatively easy to integrate in existing solution schemes. In the second level of modeling, the closure terms are directly modeled without assuming a function g a priori. Instead, the function g dependent on θ will be the outcome of the optimisation. In the third level of modeling, the outcome of the governing equations $\hat{U}$ is directly approximated. These methods give up some of the results of classical modeling schemes for turbulence, but are generally mesh-free and can work with sparse data.

In the next section, some applications of ML methods to turbulence modeling are discussed. This should represent a small collection of the current state-of-the-art, showing the diversity of the ideas and methods current under investigation.

3.2.1. State-of-the-art

One intuitive method to improve the accuracy of existing turbulence models is to optimise the model coefficients, such as the constants C_μ, $C_{\epsilon 1}$ and $C_{\epsilon 2}$ of the $k-\epsilon$ model in equation 2.20. These coefficients are often calibrated on basic flow types such as a channel flow or flat plate. However, to obtain more accurate solutions for more complex flows, it can be useful to tune the coefficients using ML. This is related to the first level of modeling in equation 3.5. For example, Edeling *et al.* calibrated the coefficients in five different turbulence models by employing Bayesian calibration using experimental data [38]. Each model was calibrated for different scenarios. They found that the coefficients vary greatly with the scenarios and that there is no single best choice of closure coefficients for the different scenarios. This immediately indicates the lack of generalisability of turbulence models. A similar Bayesian approach is used by Ray *et al.* to calibrate the coefficients of the $k-\epsilon$ model for one specific flow case, a jet flow [39]. The method showed good results for the specific case, but is unlikely to yield a general turbulence model. Furthermore, Fabrititus optimised the model coefficients of the $k-\epsilon$ and $k-\omega$ SST turbulence models by minimising the difference between experimental data and simulating data using a genetic algorithm [40]. Again, the results were largely improved for the case of which the coefficients were optimised, but not for different flow cases.

The limited successes in finding a general set of coefficients is likely originated from the fundamental assumption made in the employed turbulence models. The Boussinesq hypothesis is simply not valid in a large range of flows [21]. Hence, to improve the turbulence models, recent research focuses on deviating from the Boussinesq hypothesis. Instead of tuning constants, Tracey *et al.* tried to replace the full closure terms of the RANS equations by supervised learning algorithms [41], which is related to the second level of modeling in equation 3.5. They trained NNs on a number of CFD simulations such that they reproduced the Spalart-Allmaras RANS model. Instead of improving traditional RANS models, the potential of ML to enhance or replace the RANS models was demonstrated. This research can be seen as a pioneering research to replace traditional closure models with ML algorithms. More researchers followed to model closure terms in RANS or LES models with NNs, including [42] [43] [44] [45] [46].

Ling *et al.* were among the first to model the full Reynolds stress anisotropy tensor and thus completely bypasses the Boussinesq hypothesis; equation 2.10. After indicating the under-predictions of the anisotropic stress by RANS models, random forest regressors were trained to predict the anisotropic Reynolds stress more accurately [47]. The regressors were trained to predict the barycentric coordinates, which are derived from II and III in the Lumley triangle, on two different cases; a duct flow and flow around a wall-mounted cube. Thereafter the models were tested on a jet flow. The obtained models significantly improved the anisotropic Reynolds stress and even showed a remarkable ability to generalise across flows. However, the models were not integrated into a RANS solver and hence, they are not tested on predicting the flow velocity.

In a later study, Ling *et al.* used a deep NN to learn a model for the Reynolds stress anisotropy tensor using high-fidelity simulation data [12]. They introduced a novel architecture of the NN such

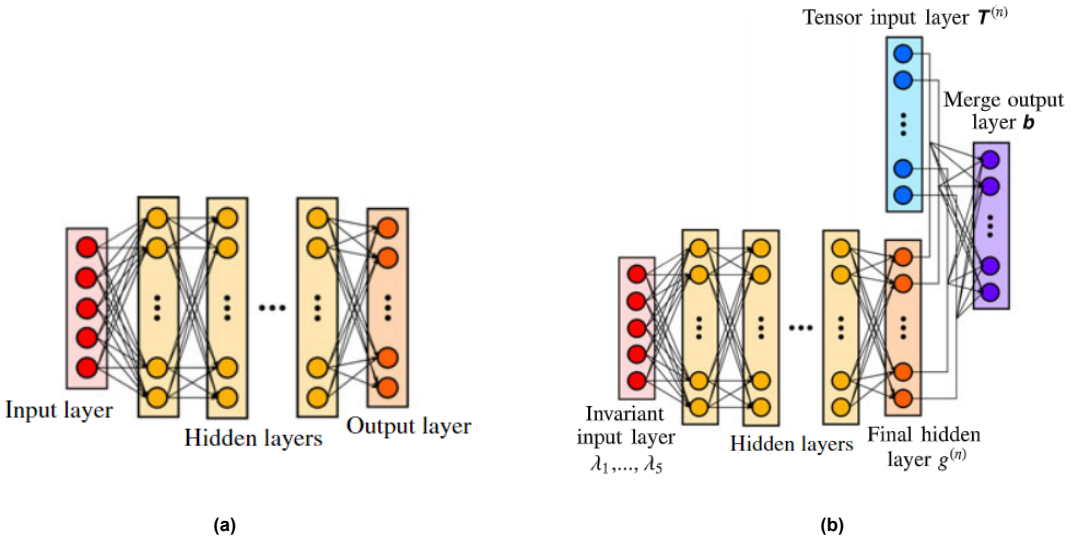


Figure 3.3: Schematic of the neural network architectures of J. Ling (b) compared to a stand neural network (a). [12, p. 159]

that physical knowledge is embedded by preserving Galilean invariance of the NN predictions. This architecture relies on the theory that the anisotropic stress tensor should lay on an invariant tensor basis. The relevant tensor basis consists of a linear combination of ten certain isotropic basis tensors $T^{(n)}$ multiplied with coefficients $g^{(n)}$, which describes the anisotropic stress as

$$b_{ij} = \sum_{n=1}^{10} g^{(n)} T^{(n)}. \tag{3.6}$$

The NN of Ling *et al.* predicts the coefficients $g^{(n)}$, which thereafter are combined with their corresponding basis tensors $T^{(n)}$. Figure 3.3b shows the structure of the Neural Network, which is called the invariant Tensor Based Neural Network (TBNN). The TBNN showed significantly more accurate predictions than a general NN, which did not embed invariance properties. A downside of using NNs is that they are difficult to integrate directly into RANS solvers due to their black-box nature and non smooth derivatives. In the research of Ling *et al.*, they first ran a default RANS simulation for which the anisotropic stresses were predicted externally. Thereafter, the velocities were updated by performing a second simulation, in which the predicted values of the anisotropic stress are inserted as a static field.

The invariant tensor basis in equation 3.6 has been used frequently in other researches, like in the research of Kaandorp and Dwight [23]. Instead of a NN, they trained a random forest to obtain the optimal coefficients as used in the linear combination of the ten tensor bases. Compared with the TBNN algorithm, the random forest was easy to implement and train. The predictions of the random forest were better than the baseline RANS simulation and performed similar compared to the TBNN algorithm. However, also a direct implementation of the random forest algorithm in a RANS solver is challenging. The anisotropic stresses were predicted externally, after which the flow field is corrected in a second RANS simulation using the static values of the anisotropic stresses.

Recent work is focused on modeling a corrective term to the Boussinesq hypothesis instead of completely replacing it. Hence, the assumption is that the Reynolds stresses can be expressed as

$$R_{ij} = -2\nu_\tau S_{ij} \ + 2k(\frac{1}{3}\delta_{ij} + b_{ij}^{\Delta}).$$

The decomposition of the anisotropy into the standard linear relation and a corrective terms is convenient for the implementation of models in RANS solvers [48]. The first reason is that some RANS solvers do not offer much freedom in modifying turbulence models, which impede the removal of the

linear term (i.e., the linear term is hidden in other expressions and not explicitly declared). Furthermore, most transport equations in the traditional turbulence models are developed intending the linear term. That is, some coefficients in the transport equations or the boundary conditions of the transport variable are calibrated considering the linear term. Hence, ignoring this term can drastically impact the numerical stability of the overall turbulence model. Finally, a mixture of implicit and explicit treatment of term is required to improve the numerical stability [13].

Wang *et al.* used random forests to reconstruct the discrepancies in RANS modeled Reynolds stresses based on DNS data [49]. They corrected not only the anisotropy but also the kinetic energy and other variables, which resulted in excelled predictive performance of the Reynolds stresses even on other unseen complex flows. However, Wang *et al.* did not propagate their model in a RANS solver. Thus the improvement of the velocities from the corrected Reynolds stress field could not be guaranteed.

Weatheritt and Sandberg used GEP to find an algebraic equation for b_{ij}^{Δ} [48] [36]. They made use of the theory that the anisotropic stress can be formed by the linear combination of invariant tensor basis and hence searched for an expression for b_{ij}^{Δ} of the form

$$b_{ij}^{\Delta} = \sum_{n=1}^{10} g^{(n)} T^{(n)}.$$

The main advantage of the methodology of Weatheritt and Sandbarg is that an explicit algebraic equation for the Reynolds anisotropic stress is obtained, which can be interpreted and directly implemented in a RANS solver. Their results showed that the framework is a viable methodology for RANS closure development. They applied the trained models to more difficult geometries, which indicates that the framework can be industrially relevant. This research provided inspiration and handles for other researches, like Zhao *et al.* and Reissmann *et al.* [50] [51]. Zhao applied GEP to LES model development and Reissmann extended the GEP method of Weatheritt and Sandberg such that models are directly integrated, trained and evaluated in CFD solvers. A drawback of the GEP method is that the implementation and training can be challenging. Complicated restrictions are required to ensure the GEP method produces valid expressions. Furthermore, GEP is known to be not well suitable for large scale problems containing much data because of their relatively high computational cost [52] [53].

Instead of GEP, the research group of Beetham *et al.* tested a similar, but simpler method to obtain an explicit algebraic equation for the correction of the anisotropic stress tensor b_{ij}^{Δ}, namely sparse symbolic regression [16]. A library was constructed in such a way that the selected candidate functions represent the coefficients $g^{(n)}$ in the linear combination of invariant tensor basis $T^{(n)}$. Around the same time, Schmelzer *et al.* also used sparse symbolic regression to find an expression for b_{ij}^{Δ}, but introduced a second correction term in the turbulence model transport equations. Since the correction of the anisotropic stress tensor will affect the turbulent transport equations k and ω, they introduced a k-corrective-frozen RANS approach. In this approach, data of both correction terms are obtained by iteratively solving the k and ω equations with high fidelity data injected for k, u and b_{ij}. Both works showed that this method generates simple and interpretable models which can directly be implemented in a RANS solver and which improve the prediction of the flow compared to default RANS simulations.

Lastly, a currently new and trendy approach to model RANS equations are Physics Informed Neural Networks (PINN's) [54], which Eivazi *et al.* introduced in turbulence modeling [55]. PINN's operate on the third level of modeling in equation 3.5. This black-box algorithm completely bypasses the closure problem by directly solving the RANS equations using an NN. As can be seen in Figure 3.4, spatial coordinates are taken as input and the full flow field variables are the output of the NN, taking the residual of the Navier Stokes equations as the loss function.

3.3. Challenges

ML found its way into turbulence modeling just recently, but major progress already has been made. However, there are still challenges and difficulties that ML-augmented turbulence models must tackle.

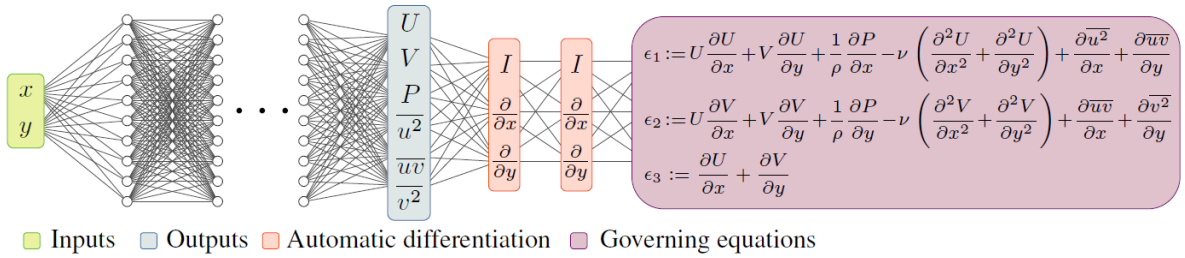


Figure 3.4: PINN used for solving the RANS equations for a general two-dimensional set-up, taken from [55, p. 2]

Some of these challenges have already been mentioned in previous sections. One of the biggest challenge is to create generalisable models, which includes the applicability of a model outside its training regime. The research of Taghizadeh *et al.* pointed out that NNs lack generalisability and will be limited in its ability to perform predictive computations of real engineering flows [56].

Secondly, it is desirable that the models are interpretable. It is unlikely that data-driven turbulence models will be ultimately successful without infusing expert human knowledge. Understanding the decisions of the ML algorithm makes it easier to improve and trust the models, for instance by including more physical knowledge. A plain NN is difficult to interpret because of its black-box methodology. Related to this is the objective to incorporate physical prior information and constraints. An obvious way to do this is by appropriately selecting the input features, which share a given property such as positivity or invariance. Alternatively, constraints to force symmetry or realisability can be explicitly included into the loss function. In general, the use of constraints can increase the stability of models, but it can also make the optimisation more difficult.

Creating robust and stable models for which the CFD simulation converges reliably is another challenge. In general, it can be very difficult to obtain converged results of a RANS simulation, let alone a modified RANS solver. Wu *et al.* observed that the RANS equations can be ill-conditioned when the modeled Reynolds stress is directly substituted into the equations [13]. Even solving the RANS equations with the Reynolds stresses from DNS data can lead to large errors in the velocities. Models that treat the Reynolds stress as an implicit source term in the RANS equations suffer less from this ill-conditioning. These include the algebraic turbulent constitutive relations discovered by symbolic regression [36].

Finally, it is favoured to use ML methods which are easy to employ. A general observation is that the best approaches not always find a widespread use, but rather those that are easy to understand and implement and are computationally cheap. Experimenting with the methods is also more attractive when they are easy to use and understand.

Reflecting these challenges on the methods found in literature as described in Sections 3.1 and 3.2, sparse symbolic regression tends to anticipate on many of these challenges. To start with the fact that sparse symbolic regression outputs an interpretable model. With sparse symbolic regression, candidate functions are selected and fitted, such that an algebraic equation is formed with a limited number of terms. For this reason, other approaches such as GEP and random forest regression increased in popularity. GEP is beneficial since it also produces an algebraic equation. Random forest regression does not generate algebraic equation as output, but it has an intuitive way of selecting features and creating models.

Furthermore, the sparse symbolic regression method is efficient, since a subset of the initial physics based candidate functions are used. Consequently, fewer forward computations are necessarily compared to other methods such as random forests and NNs, which makes the method computationally cheap. The simple algebraic equation obtained by sparse symbolic regression is also relatively easy to integrate into existing CFD solvers. NNs and random forests on the other hand are much more complicated to integrate into a solver, since the models obtained by these methods cannot be described explicitly by an easy equation. Therefore, the velocity cannot be easily computed.

Lastly, physical knowledge can be extracted from the algebraic equation which can give insight into turbulence. Furthermore, physical knowledge can be relatively easily included. For instance, Galilean invariance can be ensured by carefully constructing the library of candidate function and setting up the regression problem.

These arguments motivated to use sparse symbolic regression as method to obtain data-driven turbulence models. In the next chapter, we will discuss how sparse symbolic regression is applied in this research project.

4

Methodology

This chapter will describe the overall framework of the turbulence modeling procedure. The sparse symbolic regression method as applied in this research will be explained, followed with a description of the employed input features. Furthermore, a method to include near-wall treatment is discussed. The chapter is closed with explaining the model propagation in OpenFOAM.

4.1. Modeling Framework

We aim to develop an improved algebraic closure model for the Reynolds stress, R_{ij}, appearing in the RANS equations 2.8. The Reynolds stress tensor can be divided into its isotropic part, $\frac{2}{3}k\delta_{ij}$, and anisotropic part, $a_{ij} = 2kb_{ij}$, yielding

$$R_{ij} = a_{ij} + \frac{2k}{3}\delta_{ij} = 2k(b_{ij} + \frac{1}{3}\delta_{ij}). \tag{4.1}$$

As outlined previously, a linear eddy viscosity model based on the Boussinesq equation is frequently used to model the anisotropic part as $b_{ij} = \frac{\nu_t}{k}S_{ij}$. However, Pope recognised that a more general nonlinear representation of the anisotropic stress tensor is a finite tensor polynomial, dependent on both the normalised strain rate tensor, $\hat{S}_{ij} = \tau\frac{1}{2}(\partial_j\overline{U}_i + \partial_i\overline{U}_j)$, and the rotation rate tensor, $\hat{\Omega}_{ij} = \tau\frac{1}{2}(\partial_j\overline{U}_i - \partial_i\overline{U}_j)$, with timescale $\tau = 1/\omega$ [57]. With the use of the Cayley-Hamilton theorem, Pope derived a linear combination of ten independent tensors $T_{ij}^{(n)}$, which are symmetric and have zero trace, describing the anisotropic stress tensor as

$$b_{ij}(\hat{S}_{ij}\hat{\Omega}_{ij}) = \sum_{n=1}^{10} G^{(n)}(I_1, ..., I_5)T_{ij}^{(n)}. \tag{4.2}$$

Here, the coefficients $G^{(n)}$ may be functions of the five invariants $I_1, ..., I_5$, listed in Table 4.2. The ten basis tensors that are used in this equation are listed in Table 4.1. Using equation 4.2 has as advantage that Galilean invariance can be ensured, since the tensors form an invariant basis. This means that the anisotropy tensor is the same in different inertial frames. In this way, physical knowledge is implicitly used.

Table 4.1: Tensor bases $T_{ij}^{(n)}$ used in equation (4.2) to describe the anisotropic Reynolds stress.

$T_{ij}^{(1)} = \hat{S}_{ij}$	$T_{ij}^{(6)} = \hat{\Omega}_{ik}\hat{\Omega}_{kl}\hat{S}_{lj} + \hat{S}_{ik}\hat{\Omega}_{kl}\hat{\Omega}_{lj} - \frac{2}{3}\hat{S}_{pk}\hat{\Omega}_{kl}\hat{\Omega}_{lp}\delta_{ij}$
$T_{ij}^{(2)} = \hat{S}_{ik}\hat{\Omega}_{kj} - \hat{\Omega}_{ik}\hat{S}_{kj}$	$T_{ij}^{(7)} = \hat{\Omega}_{ik}\hat{S}_{kl}\hat{\Omega}_{lp}\hat{\Omega}_{pj} - \hat{\Omega}_{ik}\hat{\Omega}_{kl}\hat{S}_{lp}\hat{\Omega}_{pj}$
$T_{ij}^{(3)} = \hat{S}_{ik}\hat{S}_{kj} - \frac{1}{3}\hat{S}_{lk}\hat{S}_{kl}\delta_{ij}$	$T_{ij}^{(8)} = \hat{S}_{ik}\hat{\Omega}_{kl}\hat{S}_{lp}\hat{S}_{pj} - \hat{S}_{ik}\hat{S}_{kl}\hat{\Omega}_{lp}\hat{S}_{pj}$
$T_{ij}^{(4)} = \hat{\Omega}_{ik}\hat{\Omega}_{kj} - \frac{1}{3}\hat{\Omega}_{lk}\hat{\Omega}_{kl}\delta_{ij}$	$T_{ij}^{(9)} = \hat{\Omega}_{ik}\hat{\Omega}_{kl}\hat{S}_{lp}\hat{S}_{pj} + \hat{S}_{ik}\hat{S}_{kl}\hat{\Omega}_{lp}\hat{\Omega}_{pj} - \frac{2}{3}\hat{S}_{qk}\hat{S}_{kl}\hat{\Omega}_{lp}\hat{\Omega}_{pq}$
$T_{ij}^{(5)} = \hat{\Omega}_{ik}\hat{S}_{kl}\hat{S}_{lj} - \hat{S}_{ik}S_{kl}\hat{\Omega}_{lj}$	$T_{ij}^{(10)} = \hat{\Omega}_{ik}\hat{S}_{kl}\hat{S}_{lp}\hat{\Omega}_{pq}\hat{\Omega}_{qj} - \hat{\Omega}_{ik}\hat{\Omega}_{kl}\hat{S}_{lp}\hat{S}_{pq}\hat{\Omega}_{qj}$

Table 4.2: Invariants on which the coefficients $G^{(n)}$ in equation (4.2) depend.

$I_1 = \text{trace}(S^2)$	$I_2 = \text{trace}(\Omega^2)$	$I_3 = \text{trace}(S^3)$	$I_4 = \text{trace}(\Omega^2 S)$	$I_5 = \text{trace}(\Omega^2 S^2)$

We want to find an equation for the anisotropic stresses in the form of equation 4.2. However, instead of modeling b_{ij} directly, several works recommend to split the anisotropic stress tensor into a linear part, b^o, and a nonlinear part, b^Δ, and take the nonlinear part as subject of the modeling. The linear part of the anisotropic stress can be taken as a standard LEVM, such that

$$\begin{aligned} b_{ij} &= b^o_{ij} + b^\Delta_{ij} \\ &= b^\Delta_{ij} - \frac{\nu_t}{k} S_{ij}. \end{aligned} \tag{4.3}$$

Splitting the anisotropic stress tensor this way should improve stability when the model is integrated in a CFD solver. Wu *et al.* have pointed out that the reason for this is the ill-conditioning of the RANS equations [13]. Substituting Reynolds stresses with low errors from DNS databases explicitly into the RANS equations can lead to velocities with very large errors. However, partial implicit treatment of the Reynolds stresses should stabilise the RANS simulation and prevent the error amplification of the velocities.

In this research, the nonlinear part of the anisotropic stress tensor is the subject of modeling. It is calculated with equation 4.3, in which the linear part b^o_{ij} is derived from the output of the $k-\omega$ RANS model. The $k-\omega$ model computes the turbulent viscosity as $\nu_t = \frac{k}{\omega}$, where k and ω are obtained by solving the transport equations 2.17. For the complete anisotropic stress tensor b_{ij}, DNS data is used and hence, our target of modeling is derived by

$$b^\Delta_{ij} = b^{DNS}_{ij} + \frac{\nu_t^{RANS}}{k^{RANS}} S^{RANS}_{ij}, \tag{4.4}$$

in which *DNS* and *RANS* indicates that data from a DNS database or a RANS simulation is used. Hence, for each point in a specific flow case DNS and RANS data are required. The DNS data is collected from literature and the RANS data is obtained by performing a RANS simulation for the corresponding case. A more detailed explanation will be given in next chapter. The DNS data is interpolated to the spatial coordinates of the RANS data. This way of calculating the nonlinear part of the anisotropic stress, is inspired by the research of Beetham *et al.*[16] and Wu *et al.* [58]. A slightly different approach is done by Schmelzer *et al.* [15] and Weatheritt *et al.* [48]. Instead of calculating the linear part $\frac{\nu_t}{k} S_{ij}$ with RANS data, k, ν_t and S_{ij} are derived from DNS data. However, the k and ω transport equations introduce additional modeling errors in the propagating phase, even if one would perfectly fit the DNS data. By using RANS data, these errors are incorporated as well.

Our model discovery approach for the nonlinear anisotropic stress tensor b^Δ_{ij} using equation 4.2 consists of different steps and an overview of the modeling framework can be seen in Figure 4.1. First, from DNS data and the baseline RANS simulation, the target and input features are extracted. Thereafter, sparse symbolic regression is conducted to obtain an expression for b^Δ_{ij}. Lastly, the mean velocities are derived by solving the new RANS equations containing the discovered model for the anisotropic stress, which is referred to as the 'propagation' of the new Reynolds stress field to mean velocities. The next sections will explain how sparse symbolic regression is used to find an algebraic expression for the anisotropic stress tensor and which input features are used. Also a description of the model propagation is given.

4.2. Sparse Symbolic Regression

The sparse symbolic regression approach as used in this research is built upon the data-driven technique presented by Brunton *et al.*, which aims to discover the governing equations of nonlinear dynamical systems using temporally evolving data [14]. Instead of uncovering governing equations, the method is modified to identify data-driven closure models for the RANS equations. We consider the nonlinear anisotropic stress tensor and assume that it can be described by equation 4.2. This equation

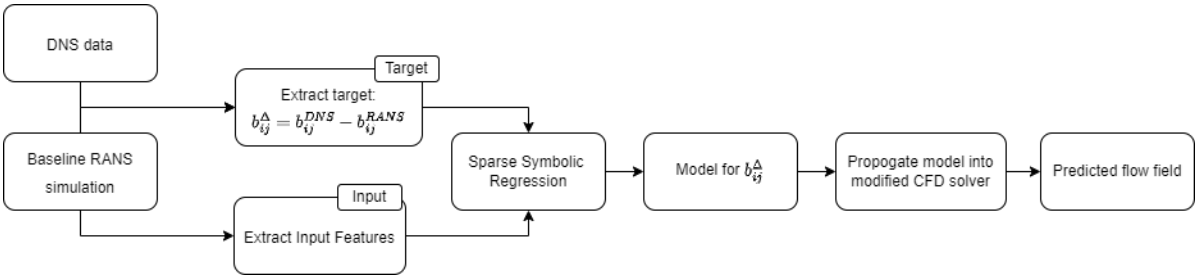


Figure 4.1: Framework of data-drive turbulence model discovery.

will be slightly extended such that it also incorporates other features besides the invariants. Furthermore, for two-dimensional flows this equation simplifies to a linear combination of only the first three basis tensors and only the first two invariants being nonzero. Hence, we aim to find an equation which describes the nonlinear anisotropic stress tensor as

$$b_{ij}^{\Delta} = \sum_{n=1}^{3} G^{(n)}(\theta) T_{ij}^{(n)},$$

where θ are the used features. To find the optimal coefficients $G^{(n)}$, a linear regression problem is set up such that

$$\boldsymbol{b}^{\Delta} = \Theta\xi. \tag{4.5}$$

In the approach of Brunton, the matrix Θ is a library of relevant candidate functions. To end up with the model of the form in equation 4.2, these candidate functions contain the invariant tensors $T_{ij}^{(n)}$, or a product of a tensor with a function of features, such as θ_1, θ_1^2 or $\theta_1\theta_2$. The candidate functions can be as complex as desired, but many candidate functions also require more computational power and memory capacity. In order to obtain a compact model for tensor $\boldsymbol{b}^{\Delta}$, it needs to be transformed into a column vector. This directly ensures that ξ is a vector as well and that the coefficients for each term in the model are scalars. Consequently, the model is rotation-invariant, which means that the same model form is guaranteed regardless of orientation. Since the anisotropic stress tensor is symmetric, the full tensor can be represented in a column vector as $\left[b_{11}^{\Delta}, b_{12}^{\Delta}, b_{13}^{\Delta}, b_{22}^{\Delta}, b_{23}^{\Delta}, b_{33}^{\Delta}\right]^T$. The same convention can be used for the library of candidate functions, since the invariant tensors $T_{ij}^{(n)}$ are symmetric as well. An example of the vector $\boldsymbol{b}^{\Delta}$ as a linear combination of a library of candidate functions is

$$\underbrace{\begin{bmatrix} b_{11}^{p=1} \\ \vdots \\ b_{11}^{p=p} \\ b_{12}^{p=1} \\ \vdots \\ b_{12}^{p=p} \\ b_{13}^{p=1} \\ \vdots \\ b_{13}^{p=p} \\ b_{22}^{p=1} \\ \vdots \\ b_{22}^{p=p} \\ b_{23}^{p=1} \\ \vdots \\ b_{23}^{p=p} \\ b_{33}^{p=1} \\ \vdots \\ b_{33}^{p=p} \end{bmatrix}}_{b^{\Delta} \in \mathcal{R}^{(p \cdot 6)}} = \underbrace{\begin{bmatrix} T_{11}^{(1),p=1} & \theta_1 T_{11}^{(1),p=1} & \dots & \theta_k T_{11}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{11}^{(1),p=p} & \theta_1 T_{11}^{(1),p=p} & \dots & \theta_k T_{11}^{(n),p=p} \\ T_{12}^{(1),p=1} & \theta_1 T_{12}^{(1),p=1} & \dots & \theta_k T_{12}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{12}^{(1),p=p} & \theta_1 T_{12}^{(1),p=p} & \dots & \theta_k T_{12}^{(n),p=p} \\ T_{13}^{(1),p=1} & \theta_1 T_{13}^{(1),p=1} & \dots & \theta_k T_{13}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{13}^{(1),p=p} & \theta_1 T_{13}^{(1),p=p} & \dots & \theta_k T_{13}^{(n),p=p} \\ T_{22}^{(1),p=1} & \theta_1 T_{22}^{(1),p=1} & \dots & \theta_k T_{22}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{22}^{(1),p=p} & \theta_1 T_{22}^{(1),p=p} & \dots & \theta_k T_{22}^{(n),p=p} \\ T_{23}^{(1),p=1} & \theta_1 T_{23}^{(1),p=1} & \dots & \theta_k T_{23}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{23}^{(1),p=p} & \theta_1 T_{23}^{(1),p=p} & \dots & \theta_k T_{23}^{(n),p=p} \\ T_{33}^{(1),p=1} & \theta_1 T_{33}^{(1),p=1} & \dots & \theta_k T_{33}^{(n),p=1} \\ \vdots & \vdots & \dots & \vdots \\ T_{33}^{(1),p=p} & \theta_1 T_{33}^{(1),p=p} & \dots & \theta_k T_{33}^{(n),p=p} \end{bmatrix}}_{\Theta \in \mathcal{R}^{(p \cdot 6) \times g}} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_g \end{bmatrix}.$$

Here, p denotes the number of data points and k the number of used features θ. In order to determine the optimal coefficient vector ξ, an ordinary least-squares problem needs to be solved, which minimises the sum of the squared residuals and can be defined as

$$\xi = \arg\min_{\hat{\xi}} ||\Theta\hat{\xi} - b^{\Delta}||_2^2. \tag{4.6}$$

To end up with a smaller subset of candidate functions, such that a simple model can be formed, sparsity-promoting regularisation is added to the least-squares optimisation problem. There are multiple variants for this, each having its own characteristics. In this research, four different sparsity promoting regression functions are used and compared, namely the LASSO regression, the Elastic net regression, the relatively new SR3 method and finally the sequential thresholding least squares problem.

4.2.1. LASSO

The objective of LASSO, which is the abbreviation of 'Least Absolute Shrinkage and Selection Operator' is to solve the minimisation problem

$$\xi = \arg\min_{\hat{\xi}} ||\Theta\hat{\xi} - b^{\Delta}||_2^2 + \lambda||\hat{\xi}||_1, \tag{4.7}$$

where the penalty term $\lambda||\hat{\xi}||_1$ represents the l_1-norm of the coefficient vector, weighted with regularisation parameter λ [59]. This l_1-norm allows only a few nonzero coefficients while shrinking the rest to zero. Larger values for λ result in more zero valued coefficients in ξ, while smaller values for λ increase the amount of nonzero coefficients. Note that when λ is set to zero, the function 4.7 becomes

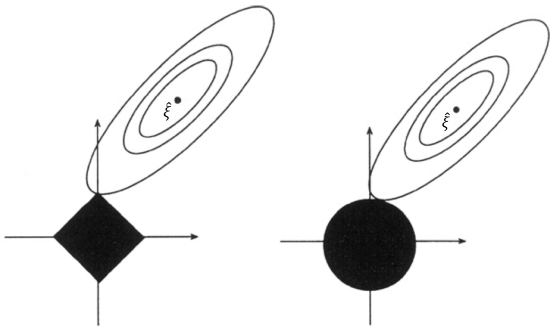


Figure 4.2: Contours of the error and constraint functions for the LASSO (left) and Ridge regression (right), obtained from [59].

the least-squares objective function.

Figure 4.2 illustrates why LASSO is able to produce coefficients that are exactly zero. The LASSO regression problem can equivalently be written as the optimisation problem

$$\begin{aligned} \text{minimize} \quad & ||\Theta\xi - b^{\Delta}||_2^2 \\ \text{subject to} \quad & ||\xi||_1 \leq t, \end{aligned} \tag{4.8}$$

where t is a tuning parameter that determines the degree of regularisation. The term $||\Theta\xi - b^{\Delta}||_2^2$ can be written as $(\xi - \hat{\xi})^T\Theta^T\Theta(\xi - \hat{\xi})$, which gives rise to an ellipse contour plot centered around the optimal solution $\hat{\xi}$, represented by the full curves in Figure 4.2. The ellipses denote regions where the sum of squared residuals take identical values. The solid square represents the constraints region. The solution of LASSO is the first place where the contours touch the square, which can occur at a corner corresponding to a zero coefficient. If the l_2-norm is taken as penalty norm instead of the l_1-norm, the constraints region can be represented by a solid sphere. This regression problem is known as Ridge regression. Since the constraints region has no corners, the contours will rarely touch the sphere on zero solutions.

Given the LASSO regression problem, we need to specify suitable values for λ. A vector

$$\lambda = [\lambda_{min}, ..., \lambda_{max}]$$

is created that defines the entire search space for which a vector $\xi^{(i)}$ is found as a solution of equations 4.7 with λ_i. We set λ_{min} to 10^{-4} and λ_{max} to 10 with a logarithmic scale between these values. The Scikit-Learn Python package [60] is used to solve the LASSO problem, which employs a coordinate descent algorithm to fit the coefficients. Such an algorithm minimises the objective function along one direction (one coefficient in ξ) at a time, and iteratively cycles through all the directions in ξ until convergence is reached. More details of the coordinate gradient descent algorithm can be found in Appendix A.

4.2.2. Elastic net

Another sparsity-promoting regularisation of the least-squares problem is the Elastic net (Enet) problem. The Enet regression problem is an extension of the LASSO by inserting an additional l_2-norm regularisation term resulting in the minimisation problem

$$\xi = \arg\min_{\hat{\xi}} ||\Theta\hat{\xi} - b^{\Delta}||_2^2 + \lambda\rho||\hat{\xi}||_1 + 0.5\lambda(1-\rho)||\hat{\xi}||_2^2. \tag{4.9}$$

Here, the l_2-norm, known from Ridge regression, enforces the coefficients to be small without setting them to zero. This introduces the ability to identify also correlated candidate functions instead of picking a single one. In equation 4.9, $\rho \in [0, 1]$ is a mixing parameter between the l_1- and l_2-norm. As search space the values for λ are used as in the LASSO regressor. For ρ we use

$$\rho = [0.01, 0.1, 0.2, 0.5, 0.7, 0.9, 0.95, 0.99, 1.0].$$

The problem is again solved using the Scikit-Learn package, performing the coordinate descent algorithm.

4.2.3. Sequential Thresholding Least Squares

Brunton *et al.* presented a Sequential Thresholded Least-Squares (STLSQ) algorithm [14]. This algorithm starts by finding a solution of the Ridge regression problem

$$\xi = \arg\min_{\hat{\xi}} ||\Theta\hat{\xi} - b^{\Delta}||_2^2 + \lambda||\hat{\xi}||_2^2. \tag{4.10}$$

All the coefficients in ξ below a given threshold are set to zero. Subsequently, with the indices of the remaining non-zero coefficients a new solution of the Ridge regression problem is obtained. These new coefficients are again thresholded and this procedure is repeated until the nonzero coefficients converged and non of them are below the threshold. According the paper, this algorithm is computationally efficient and it rapidly converges to a sparse solution. Furthermore, it is robust to noise and benefits from simplicity. Only the threshold parameter determines the degree of sparsity in ξ.

To define our search space we vary λ with the values $[0.01, 0.05, 0.1, 0.5]$ and use as threshold values between 10^{-3} and 10, which are logarithmically scaled. The PySINDy package is used to perform STLSQ regression, in which the Ridge regression is solved using the Cholesky decomposition [61] [62]. More details about the Cholesky decomposition can be found in Appendix B.

4.2.4. Sparse Relaxed Regularised Regression

Sparse Relaxed Regularised Regression (SR3) [63] attempts to minimise the objective function

$$\xi = \arg\min_{\hat{\xi},u} \frac{1}{2}||\Theta\hat{\xi} - b^{\Delta}||_2^2 + \lambda R(u) + \frac{1}{2\nu}||\xi - u||_2^2. \tag{4.11}$$

Here, the auxiliary variable u is introduced to add relaxation to the optimisation function. $R(u)$ is a regulariser and λ is a hyperparameter that determines the strength of this regularisation. So, taking $R(\cdot) = ||\cdot||_1$, a relaxed version of LASSO is recovered. However, we will use no additional regulariser $R(\cdot)$ but apply hard thresholding on the coefficients. This makes the regression similar to thresholded least squares. Hyperparameter ν determines the level of relaxation. Decreasing ν encourages u and ξ to be close, whereas increasing ν allows the regularised coefficients ξ to be farther from u. According [64], the SR3 approach should be more generalisable to new problems than the STLSQ regression method.

As search space we set ν as $[0.01, 0.1, 1, 10]$ and used for the thresholds the same thresholds as used in the STLSQ regression. The problem is solved using the PySINDy package, in which the Cholesky decomposition is performed.

4.2.5. To Normalise or not to Normalise?

Before solving the optimisation functions as introduced in previous section, the candidate functions of the library can be normalised. Schmelzer *et al.* used this approach, since they suggest that the relevance of each candidate function should not depend on its magnitude [15]. When we normalise the candidate functions and solve the regression functions, the best fitting functions are selected. However, to obtain a model with non-normalised functions, an additional regression is done on the selected candidate functions in original non-normalised forms. This is required for the implementation of the model in the CFD solver. The additional regression to inference the selected candidate functions is done by using the Ridge regression

$$\xi = \arg\min_{\hat{\xi}} ||\Theta^s\hat{\xi} - b^{\Delta}||_2^2 + \lambda_r||\hat{\xi}||_2^2. \tag{4.12}$$

Here, the elements of Θ associated with the inactive candidates are excluded from the library to obtain Θ^s and thus, are not modified during this regression step. The l_2-norm regularisation term $\lambda_r||\hat{\xi}||_2^2$,

causes the magnitude of the coefficients to shrink, but cannot be set to zero. In general, high values for the weighting parameter λ_r lead to a lower magnitude of the coefficients. This could be beneficial since previous research reported that a CFD solver, in which the models will be implemented, have difficulties in producing a converged solution for models with large coefficients [36], [48]. A value of 0.1 for λ_r seemed to produce reasonable models. The Ridge regression is solved with the Skicit Learn package, performing the Cholesky decomposition.

The Ridge regression can introduce modeling and numerical errors. Therefore, the question is if it is necessary to normalise the library beforehand or not. When the library is not normalised, the results can directly be used and propagated. However, solving the sparse regression problems for different values of the hyperparameters can result in similar functions with the same active candidate functions. In that case, the model is selected with the best training performance.

4.2.6. Adding Constraints

As explained in Section 2.4.1, the anisotropic stress should be between certain values to be a realisable flow. The diagonal components should be in the interval $[-\frac{1}{3}, \frac{2}{3}]$, and the off-diagonal components in the interval $[-\frac{1}{2}, \frac{1}{2}]$. However, the training procedure does not take this into account and hence can result in unrealisable flows. To include this physical knowledge during training, constraints are added to the STLSQ and SR3 regression problems. Considering the STLSQ optimisation function, the problem evolves into

$$\begin{aligned} &\min_{\hat{\xi}} ||\Theta\hat{\xi} - b^{\Delta}||_2^2 + \lambda||\hat{\xi}||_2^2 \\ &s.t. \quad a_1 \leq (b^0 + \Theta\xi) \leq a_2. \end{aligned} \tag{4.13}$$

Here, a_1 and a_2 are column vectors containing the values of the intervals for the flattened anisotropic stress tensor. The problem is solved with the CVXOPT Quadratic Programming solver [65].

Adding constraints to the SR3 problem looks similar. However, instead of hard thresholding the regression problem as done in the SR3 problem without constraints, the constrained problem contains a l_1-norm regularisation without hard thresholding. The problem to be solved is

$$\begin{aligned} &\min_{\hat{\xi}} 0.5||\Theta\hat{\xi} - b^*||_2^2 + \lambda||u||_1 + \frac{0.5}{\nu}||\hat{\xi} - u||_2^2 \\ &s.t. \quad a_1 \leq (b^0 + \Theta\xi) \leq a_2. \end{aligned} \tag{4.14}$$

The l_1-norm is applied since the used PySINDy package does not allow to use inequality constraints in combination with the hard thresholding method. Within PySINDy, the problem is solved with the Embedded Conic Solver (ECOS) using the CVXPY package [66] [67]. The question is whether the added constraints result in models which are realisable, and if this has an effect on the generalisability and robustness of the models. Unfortunately, we did not manage to develop a working solving scheme for the constrained LASSO and Enet regression problems, mainly because of time limitations.

4.3. Input Features

Using the right input features can have a significant influence on the performance of the learned model. In the nonlinear eddy viscosity hypothesis of Pope, the coefficients are functions of the invariants of the strain and rotation rate tensors, given in Table 4.2. However, it is possible to include additional features to predict the coefficients of the basis tensors. With raw local flow variables, such as the mean pressure $\overline{p}$, mean velocity $\overline{U}$, turbulent kinetic energy k, turbulent dissipation rate ϵ, the eddy viscosity ν_t and nearest wall distance d, Ling and Templeton constructed a rich set of input features based on domain knowledge and physical intuition [47]. From this set we selected seven features, which in our opinion are applicable to use in this research. These features are listed in Table 4.3, including a short description. The features are coordinate-free and should be non-dimensional. Non-dimensionalising the inputs is done according the formula

$$q_\beta = \frac{\tilde{q}_\beta}{|\tilde{q}_\beta| + |q_\beta^*|}. \tag{4.15}$$

Here, $\tilde{q}_\beta$ are the raw values of the features and q^*_β are the corresponding normalisation factors. This formulation is preferred over the non-dimensionalising with formula $\frac{\tilde{q}_\beta}{|q^*_\beta|}$, since it reduced the probability of the denominator approaching zero and constraints the input to lie in the range $(-1, 1)$ for $|q^*_\beta| > 0$.

To study the capabilities of the physical features, two input libraries are used. The first library only consists of functions of the invariants. These functions also include higher orders and combinations of the (higher order) invariants. This library will be referred to as 'invariant library' in the rest of this research. The construction of this library is done as follows. As primitive input features the invariants I_1 and I_2 are used, since only two-dimensional flow cases are considered. In addition, the second and third power of the invariants are included and the resulting candidates are multiplied with each other. This results in the vector

$$\mathcal{B} = [1, I_1, I_2, I_1^2, I_2^2, I_1^3, I_2^3, I_1 I_2, I_1^2 I_2^2, I_1^2 I_2, I_1^3 I_2, ...]. \tag{4.16}$$

Note that vector $\mathcal{B}$ also contains 1, to include a constant function c. In total, vector $\mathcal{B}$ consists of 41 functions having a maximum power of seven. Finally, each function in $\mathcal{B}$ is multiplied with each of the three base tensor $T^{(n)}_{ij}$, which results in the library

$$\Theta_I = [T^{(1)}_{ij}, T^{(2)}_{ij}, ..., I_1^2 I_2^2 T^{(3)}_{ij}]. \tag{4.17}$$

In order to avoid candidate functions with too small values, functions are excluded if the norm of their column vector is lower than 10^{-3}. This results in total number of 22 candidate functions, in which the invariants have a maximum power of three.

The second library will also include the physical features from Table 4.3. In this library, only single orders of all the features are used and they are not multiplied with each other. Hence, the final library consists of 30 functions and reads

$$\Theta_{II} = [T^{(1)}_{ij}, I_1 T^{(1)}_{ij}, I_2 T^{(1)}_{ij}, q_1 T^{(1)}_{ij}, ..., q_7 T^{(3)}_{ij}]. \tag{4.18}$$

Table 4.3: Additional physical features used for creating the library of candidate functions of equation 4.18. The features are obtained from [47].

Index	Features	Normalisation	Comment
q_1	$\frac{1}{2}(\|\|R\|\|^2 - \|\|S\|\|^2)$	$\|\|S\|\|^2$	Ratio of rotation rate to strain rate
q_2	k	$\frac{1}{2}\overline{u}_i\overline{u}_i$	Ratio of the turbulent kinetic energy to the mean kinetic energy
q_3	$\overline{u}_i \frac{\partial p}{\partial x_k}$	$\sqrt{\frac{\partial p}{\partial x_j}\frac{\partial p}{\partial x_j}\overline{u}_i\overline{u}_i}$	Pressure gradient along streamline
q_4	k/ϵ	$\frac{1}{\|\|S\|\|}$	Ratio of the turbulent time scale to the mean flow time scale
q_5	$\overline{u}_i \frac{\partial k}{\partial x_i}$	$\|\overline{u'_j u'_k} S_{jk}\|$	Ratio of convection to production of TKE
q_6	$\sqrt{\frac{\partial p}{\partial x_i}\frac{\partial p}{\partial x_i}}$	$\frac{1}{2}\rho\frac{\partial \overline{u}_k^2}{\partial x_k}$	Ratio of pressure normal stresses to shear stresses
q_7	$\|\overline{u}_i\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j}\|$	$\sqrt{\overline{u}_i\overline{u}_i\overline{u}_i\frac{\partial \overline{u}_i}{\partial x_j}\overline{u}_k\frac{\partial \overline{u}_k}{\partial x_j}}$	Non-orthogonality between velocity and its gradient

4.4. Damping

As described in Section 2.5, the region near the wall has a lot of influence on the inner part of a channel flow. Flow variables scale differently in the near-wall region than in the inner part and therefore, it is difficult to find a model which satisfies both the near-wall and inner region. This is a challenge of

turbulence modeling in general. To improve the behaviour in the near-wall region of their model, Wallin and Johansson used the van Driest function

$$f = 1 - \exp\left(-\frac{y^+}{A}\right), \tag{4.19}$$

where y^+ is the dimensionless wall distance which can be calculated with equation 2.31, and A is a constant with a value of 26. Coefficient A is calibrated on DNS data. Wallin and Johansson aimed to develop an explicit algebraic Reynolds stress turbulence model [5]. As basis they used the theorems of Pope and Caley-Hamilton to describe the anisotropic stress tensor in the form of equation 4.2 and derived analytically the coefficients in front of the tensors. For a two-dimensional solution, they ended up with the following model for the anisotropy including the near-wall van Driest damping function

$$b = f\beta_1 T^{(1)} + \left(f^2\beta_2 - (1-f^2)\frac{B_2}{2\max(I_1, I_1^{eq})}\right)T^{(2)} + (1-f^2)\frac{3B_2 - 4}{\max(I_1, I_1^{eq})}T^{(3)}, \tag{4.20}$$

where the β's denote the coefficients and B_2 and I_1^{eq} are constants. Based on DNS data, a value of 1.8 was used for B_2.

Inspired by equation 4.20, models are created including the Van Dries damping function. Two modeling frameworks are constructed to obtain such models. In the first framework, damping is added after discovering a base model. This part serves as a proof of concept such that base models are discovered only using the normalised invariant library and by solving the constrained STLSQ problem. Furthermore, the models are trained on the damped anisotropic stresses $\tilde{b}_{ij}^{\Delta} = f b_{ij}^{\Delta}$. This is done to train a model only on the inner region of the flow. When a model is discovered the damping term will be added as follows

$$\begin{aligned}
b_{11} &= f(b_{11}^o + b_{11}^{\Delta}) + (1-f)(0.9 - \frac{1}{3}),\\
b_{22} &= f(b_{22}^o + b_{22}^{\Delta}) + (1-f)(-\frac{1}{3}),\\
b_{33} &= f(b_{33}^o + b_{33}^{\Delta}) + (1-f)(\frac{2}{3} - 0.9),\\
b_{12} &= f(b_{12}^o + b_{12}^{\Delta}),\\
b_{13} &= f(b_{13}^o + b_{13}^{\Delta}),\\
b_{23} &= f(b_{23}^o + b_{23}^{\Delta}).
\end{aligned} \tag{4.21}$$

The additional terms for the diagonal components will cause the anisotropic stresses to reach a specific asymptotic value at the wall. These values are based on DNS simulations of a plain channel flow. However, by including the damping function in this way, the coordinate-free form of the damping function will be lost. This can lead to errors in curved domains.

In the second modeling framework, damping is added to the library such that models directly contain damping. The damped library consists of the invariant library multiplied by the functions $f, f^2, 1-f$ and $1-f^2$. To avoid unrealisable flows, the constrained STLSQ regression problem is used to discover the models.

4.5. Propagation of the Anisotropic Stress Tensor

To assess the performance of the learned model over the standard LEVM, the learned models have to be implemented in a CFD solver. The CFD solver solves the RANS equations in combination with a specific turbulence model such as the $k-\omega$ model. Besides improving the accuracy in describing the Reynolds stresses, the ultimate goal is that the learned models improve the prediction of the velocity field. In this project, the open source software OpenFOAM is used [68]. The integration of the learned models in OpenFOAM should also expose the models' shortcomings, such as sensitivity or stability issues in the CFD solver.

4.5.1. SIMPLE Algorithm

In OpenFOAM, the simpleFoam solver is used to solve the steady-state incompressible RANS equations. These equations, without any body forces, can be written as

$$\overline{U}_j \frac{\partial \overline{U}_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[-\overline{p} + \nu\left(\frac{\partial \overline{U_i}}{\partial x_j} + \frac{\partial \overline{U}_j}{\partial x_i}\right) - R_{ij}\right]. \tag{4.22}$$

SimpleFoam uses the SIMPLE (Semi-Implicit-Method for Pressure Linked Equations) algorithm to solve these equations [69]. In this algorithm, the equations for each variable characterising the system, which are the velocity $\overline{\boldsymbol{U}}$, the pressure $\overline{p}$ and the variables characterising turbulence, are solved sequentially and the solutions of the preceding iteration are inserted in the subsequent iteration [70]. The process starts by linearising the momentum equations and discretising all the terms in space. When the discretised terms are grouped together, for each velocity component $\overline{U}_j$ a matrix equation can be obtained of the form

$$A^{m-1}\overline{U}_j^m = Q_1^{m-1} - G_i(\overline{p}^m), \tag{4.23}$$

where A represents the matrix coefficients from the discretised convection and diffusion terms of the momentum equation and G_i denotes the i^{th}-component of the gradient operator. The source term Q contains any body force or other terms that can be explicitly computed in terms of $\overline{U}_i^{m-1}$. Thereafter, the matrix A will be split into a diagonal part A_D and off-diagonal part A_{OD} resulting in

$$(A_D + A_{OD})\overline{U}_i^* = Q - G_i(\overline{p}^{m-1}). \tag{4.24}$$

This equation is solved using the pressure from the previous iteration. Because of writing convenience, the superscript m denoting values of the current iteration is omitted. In general, the velocity field obtained by solving this equation will not satisfy the continuity equation. Therefore, both the pressure and velocities need to be corrected by

$$\overline{p}^* = \overline{p}^{m-1} + \overline{p}', \qquad \overline{U}_i^{**} = \overline{U}_i^* + \overline{U}_i'. \tag{4.25}$$

The corrected velocity and pressure should satisfy the following simplified version of equation 4.23

$$A_D U_i^{**} + A_{OD}\overline{U}_i^* = Q - G_i(\overline{p}^*). \tag{4.26}$$

Here, the corrected velocity is only applied to the diagonal part A_D. Subtracting equation 4.24 from equation 4.26 gives us a relation between the velocity and pressure corrections

$$A_D U_i' = -G_i(\overline{p}') \qquad \longrightarrow \qquad \overline{U}_i' = -(A_D)^{-1}G_i(\overline{p}'). \tag{4.27}$$

Since the diagonal matrix can be easily inverted, this relation can be solved efficiently. If $\overline{U}_i^{**}$ was also applied to the off-diagonal matrix in equation 4.26, the equation would be too complicated to derive a relation between the two corrections. However, neglecting the effect of velocity corrections in the off-diagonal terms causes the algorithm to converge slowly or not at all. The convergence can be improved if only a portion of $\overline{U}'$ and $\overline{p}'$ is added to $\overline{U}_i^*$ and p^{m-1} respectively, which is called under-relaxation. This is controlled by the under-relaxation parameter α used in the momentum and pressure-correction equations. For instance, instead of equation 4.25 the pressure-correction equation is computed as

$$\overline{p}^* = \overline{p}^{m-1} + \alpha_p \overline{p}', \tag{4.28}$$

where $0 \leq \alpha_p \leq 1$. Finally, the correction equation for the pressure is obtained. It is required that the corrected velocities $\overline{U}_i^{**}$ satisfy the discretised continuity equation

$$D(\rho\overline{\boldsymbol{U}}^*) + D(\rho\overline{\boldsymbol{U}}') = 0. \tag{4.29}$$

Using expression 4.27, the equation for the pressure correction is

$$D(\rho(A_D)^{-1}G(\overline{p}')) = D(\rho\overline{\boldsymbol{U}}^*). \tag{4.30}$$

Once this equation is solved, the velocities and pressure are updated using equations 4.27 and 4.25. These are the solutions of iteration m and a new iteration will start.

In summary, the outer iteration loop m starts with the latest solutions $\overline{U}_i^n$ and $\overline{p}^n$ as estimates for $\overline{U}_i^{n+1}$ and $\overline{p}^{n+1}$. Then, a solution for $\overline{U}_i^*$ is obtained by solving the linearised momentum equations. Thereafter, the correction for the pressure is solved giving p', with which the velocities and pressure are corrected to obtain $\overline{p}^*$ and $\overline{U}_i^{**}$ satisfying the continuity equation. At last, additional transport equations need to be solved, such as the turbulence quantities. This is repeated until all corrections are negligibly small.

4.5.2. Modified simpleFoam Solver

This section describes how the discovered models are implemented in OpenFOAM. To propagate the model of the anisotropy tensor into the flow field, a custom solver is implemented in OpenFOAM using the simpleFoam solver as a baseline. In the source code of simpleFoam, the momentum equation without the pressure term is defined as follows:

```
tmp<fvVectorMatrix> tUEqn
    (
        fvm::div(phi, U)
      + MRF.DDt(U)
      + turbulence->divDevReff(U)
    ==
        fvOptions(U)
    );
```

Here, the term `turbulence->divDevReff(U)` calculates

$$\frac{\partial}{\partial x_j}\left((\nu+\nu_t)\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\right). \tag{4.31}$$

This includes a part of the Reynolds stress as modeled by the Boussinesq hypothesis. The Boussinesq assumption models the Reynolds stress, R_{ij}, as

$$R_{ij} = 2k(b_{ij}+\frac{1}{3}\delta_{ij}) = -\nu_t\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)+\frac{2}{3}k\delta_{ij}.$$

Note that the term $\frac{2}{3}k\delta_{ij}$ is not included by `turbulence->divDevReff(U)` but will be included in the pressure gradient term. To the equation above we want to add the corrective term for the anisotropic stress, such that the Reynolds stresses are modeled as $2k(b_{ij}+b_{ij}^{\Delta}+\frac{1}{3}\delta_{ij})$. In the simpleFoam solver, we replace `turbulence->divDevReff(U)` by the direct calculations of the Boussinesq hypothesis and b_{ij}^{Δ}:

```
tmp<fvVectorMatrix> tUEqn
    (
        fvm::div(phi, U)
      + MRF.DDt(U)
      - fvc::div((turbulence->nuEff())*dev(T(fvc::grad(U))))
      - fvm::laplacian(turbulence->nuEff(), U)
      + fvc::div(dev(2.0*turbulence->k()*bDelta_))
     ==
        fvOptions(U)
    );
```

Here, the term `turbulence->nuEff()` calculates $(\nu+\nu_t)$, where ν_t and k are computed by the chosen turbulence model. Since the computation of k and ν_t depend on the Reynolds stress as well, the respective turbulence models have to be modified too. When using the $k-\omega$ equations, the production term $\mathcal{P} = -(R_{ij})\frac{\partial U_i}{\partial x_j}$ in those equations has to be modified to

$$\mathcal{P} = -(2kb_{ij}+2kb_{ij}^{\Delta}+\frac{2}{3}k\delta_{ij})\frac{\partial U_i}{\partial x_j}. \tag{4.32}$$

The $k-\omega$ equations in the turbulence model then look like:

```
#include "Model.H"
// Calculate (- 2*k*b / nu_t) * grad(U) = (2S) * grad(U)
const volScalarField::Internal G1byNu
    (
        gradU().v() && dev(twoSymm(tgradU().v()))
    );

// Calculate  (2*k*b_Delta/nu_t) * grad(U) = (2*Omega*b_Delta) * grad(U)
volScalarField G2byNu = symm(tgradU()) && (2.0*omega_*bDelta_);

// Calculate 2*k*(b+b_Delta)/nu_t
const volScalarField::Internal GbyNu
    (
        G1byNu - G2byNu
    );

// Calculate R = 2*k*(b+b_Delta)
const volScalarField::Internal G(this->GName(), nut()*GbyNu);

// Turbulence specific dissipation rate (Omega) equation
tmp<fvScalarMatrix> omegaEqn
(
    fvm::ddt(alpha, rho, omega_)
  + fvm::div(alphaRhoPhi, omega_)
  - fvm::laplacian(alpha*rho*DomegaEff(), omega_)
 ==
    gamma_*alpha()*rho()*G/(nut+nutSmall)
  - fvm::SuSp(((2.0/3.0)*gamma_)*alpha()*rho()*divU, omega_)
  - fvm::Sp(beta_*alpha()*rho()*omega_(), omega_)
  + fvOptions(alpha, rho, omega_)
);
solve(omegaEqn);

// Turbulent kinetic energy equation
tmp<fvScalarMatrix> kEqn
(
    fvm::ddt(alpha, rho, k_)
  + fvm::div(alphaRhoPhi, k_)
  - fvm::laplacian(alpha*rho*DkEff(), k_)
 ==
    alpha()*rho()*G
  - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
  - fvm::Sp(Cmu_*alpha()*rho()*omega_(), k_)
  + fvOptions(alpha, rho, k_)
);
solve(kEqn);
```

5

Test Cases

To train and test algebraic models for the nonlinear anisotropic stress, the open-source dataset developed by McConkey *et al.* is used [71]. This data set contains a variety of RANS simulations with matching DNS or LES data, which are available. From this data set three cases are selected having similar flow characteristics, which are a channel with periodic hills (PH), a converging-diverging (CD) channel and a curved backward facing step (CBFS). For each case, the $k-\omega$ RANS simulation is performed in OpenFOAM using a standardised set of numerical schemes. A second-order upwind scheme is used for discretising the convective term $\rho(\overline{U}\cdot\nabla)U_i$ in the momentum equations. A first-order upwind scheme is used for discretising the convective terms $\overline{U}\cdot\nabla k$ and $\overline{U}\cdot\nabla\omega$ in the turbulence transport equations. For the diffusive terms $\nabla^2\overline{U_i}$, $\frac{\partial}{\partial x_j}\overline{U_i'U_j'}$, $\nabla^2 k$ and $\nabla^2\omega$ a second-order central difference scheme is used. The governing equations are solved with the simpleFoam solver. In this solver, the Generalised Geometric Algebraic Multigrid (GAMG) solver is used for solving the pressure equation and the Preconditioned Bi-Conjugate Gradient (PBiCGStab) solver is used for the velocity equations and the $k-\omega$ equations. For each case, 15.000 points are used to train the models. A random sampling is applied on the points of the CD and CBFS cases. For the PH case, this includes all the points of the domain.

5.1. Periodic Hill

A common benchmark problem for turbulence modeling is a flow over periodic hills. These periodic hills cause the flow to separate behind the descending slope, which is challenging for RANS models to capture accurately. Highly accurate data is provided by Xiao *et al.*, who performed DNS of a flow over series of periodic hills at Reynolds number $Re = 5.600$ [72]. This Reynolds number is based on crest height and the bulk velocity at the crest. Cyclic boundary conditions are applied in the streamwise direction, and no-slip boundary conditions are applied at the top and bottom walls. The spanwise direction is homogeneous, which makes this case a two-dimensional problem. The same geometry and boundary conditions are implemented in OpenFOAM to perform a RANS simulation, where empty boundary conditions are used for the $x-y$ planes. Figure 5.1 shows a schematic of the flow geometry and a contour plot of the RANS predicted velocity. The structured hexahedral mesh consists of 100 and

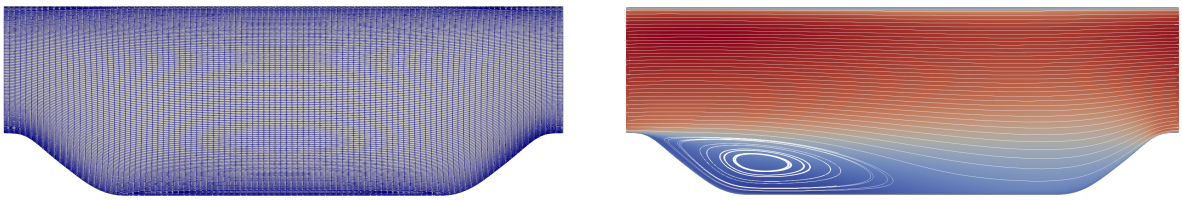


Figure 5.1: Structured hexahedral mesh and velocity field obtained from RANS simulation of the periodic hill, $Re = 5.600$

Table 5.1: Boundary and initial conditions used in the $k-\omega$ RANS simulation of the periodic hill shown in Figure 5.1.

	Inlet	Outlet	Walls	Sides	Internal field
U	Cyclic	Cyclic	No Slip	Empty	$0.028\ m/s$
p	Cyclic	Cyclic	zeroGradient	Empty	$0.0\ m^2/s^2$
k	Cyclic	Cyclic	kLowReWallFunction	Empty	$6 \cdot 10^{-8}\ m^2/s^2$
ω	Cyclic	Cyclic	omegaWallFunction	Empty	$1.5 \cdot 10^{-5}\ s^{-1}$
ν_τ	Cyclic	Cyclic	nutLowReWallFunction	Empty	$0.0\ m^2/s$

150 cells in the x and y direction, respectively. The mesh is stretched in the vertical direction, having a refinement towards the lower and upper walls. As baseline RANS simulation we used the $k-\omega$ model with the boundary and initial conditions as indicated in Table 5.1. In this table, the boundary conditions *'kLowReWallFunction'*, *'omegaWallFunction'* and *'nutLowReWallFunction'* indicate that wall functions are applied as explained in Section 2.5. The internal field adapts to the applied boundary conditions, leading to a steady-state velocity profile.

5.2. Converging Diverging Channel

Laval and Marquielle performed a DNS simulation of a flow within a channel in which an asymmetric bump is placed, exposed to an adverse pressure gradient for $Re = 12.000$ [73]. This Reynolds number is based on the maximum inlet velocity and the channel half-height. At the inlet, a fully developed channel flow enters the domain. The flow accelerates at the converging part of channel and decelerates again at the diverging part. The diverging part causes the flow to mildly separate, which causes a high turbulent kinetic energy zone right after the bump ahead of the separated region. The geometry of the CD channel as used in the RANS simulation can be seen in Figure 5.2. The structured mesh consists of 564 and 175 cells in the x and y direction, respectively. A refinement of the mesh is applied at the lower and upper walls. The boundary conditions, noted in Table 5.2, are set to generate a maximum velocity of $U_{max} = 1.0\ m/s$ to match the DNS simulation. To obtain the RANS inlet condition, a flat version of the domain was simulated to produce a fully developed flow.

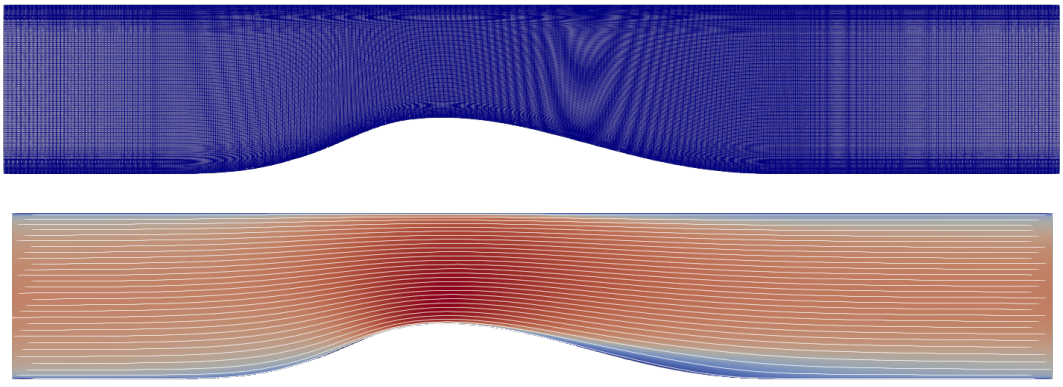


Figure 5.2: Structured hexahedral mesh and velocity field obtained from RANS simulation of the converging diverging channel, $Re = 12.600$

Table 5.2: Boundary and initial conditions used in the $k-\omega$ RANS simulation of the converging diverging channel shown in Figure 5.2.

	Inlet	Outlet	Walls	Sides	Internal field
U	Fully-developed, $\bar{U} = 1.0\ m/s$	zeroGradient	No Slip	Empty	$0.845\ m/s$
p	zeroGradient	$p = 0$	zeroGradient	Empty	$0.0\ m^2/s^2$
k	Fully-developed	zeroGradient	kLowReWallFunction	Empty	$4.3 \cdot 10^{-4}\ m^2/s^2$
ω	Fully-developed	zeroGradient	omegaWallFunction	Empty	$0.27\ s^{-1}$
ν_τ	calculated;	calculated;	nutLowReWallFunction	Empty	$0.0\ m^2/s$

Table 5.3: Boundary and initial conditions used in the $k-\omega$ RANS simulation of the curved backward facing step shown in Figure 5.3.

	Inlet	Outlet	Walls	Sides	Internal field
U	Fully-developed, $\bar{U} = 1.0\ m/s$	zeroGradient	No Slip	zeroGradient	$0.845\ m/s$
p	zeroGradient	$p = 0$	zeroGradient	zeroGradient	$0.0\ m^2/s^2$
k	Fully-developed	zeroGradient	kLowReWallFunction	zeroGradient	$6.0 \cdot 10^{-4}\ m^2/s^2$
ω	Fully-developed	zeroGradient	omegaWallFunction	zeroGradient	$4.4 \cdot 10^{-2}\ s^{-1}$
ν_τ	calculated	calculated	nutLowReWallFunction	zeroGradient	$0.0\ m^2/s$

5.3. Curved Backward Facing Step

For the flow over a Curved Backward-Facing Step (CBFS) with $Re = 13.700$ LES data was used as reference data simulated by Bentaleb *et al.* [74]. The Reynolds number is based on the mean inlet velocity. As well as the other cases, the CBFS case features separation of a fully developed flow. The geometry as used in the RANS simulation can be seen in Figure 5.3. The structured mesh consists of 712 and 104 cells in the x and y direction, respectively. The boundary conditions are shown in Table 5.3. To develop the inlet boundary conditions of a fully developed flow with $U_{max} = 1.0$ m/s, the same procedure was used as done for the CD case.

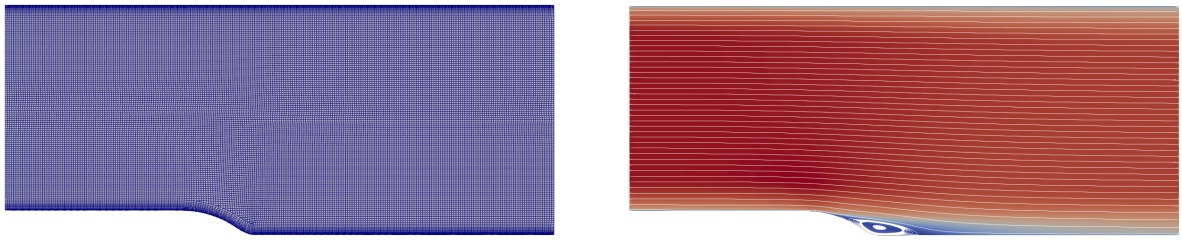


Figure 5.3: Structured hexahedral mesh and velocity field obtained from RANS simulation of the curved backward facing step, $Re = 13.700$

6

Results

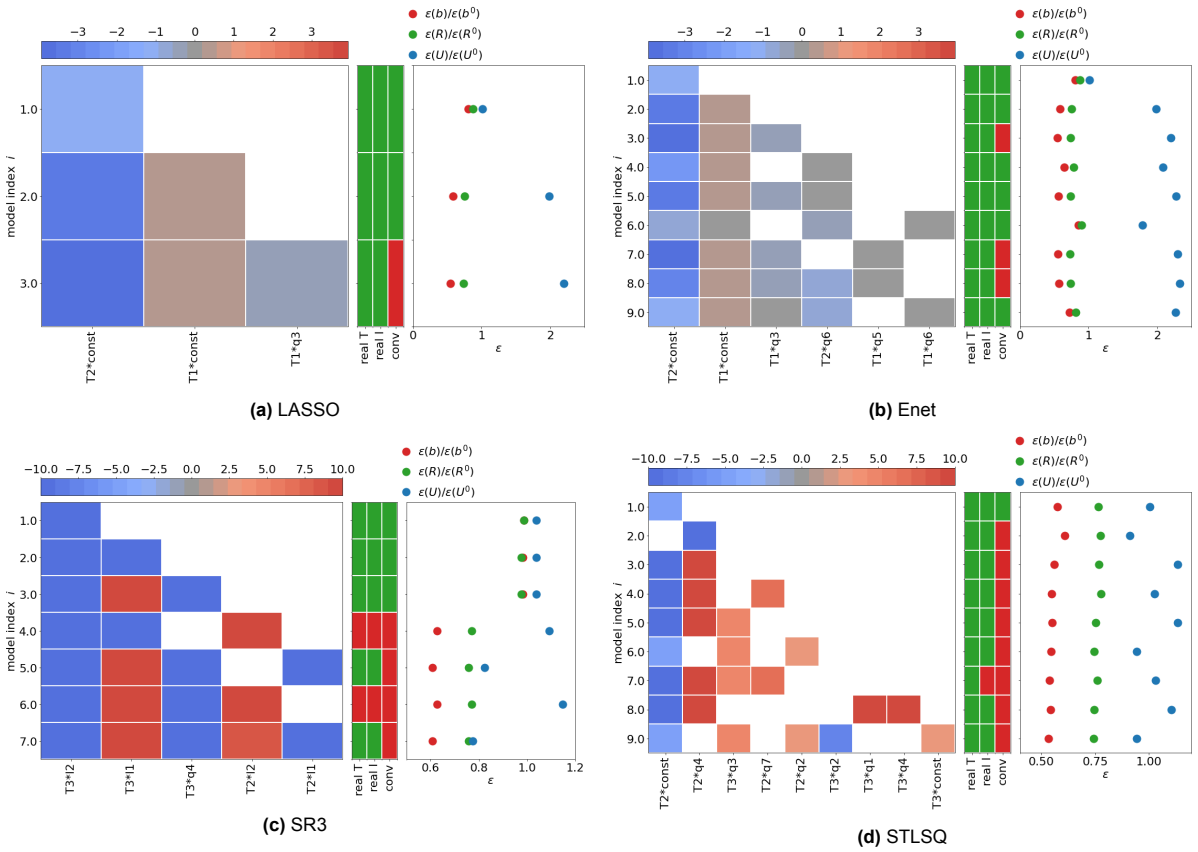
This chapter discusses the results obtained in this project. First, the best performing discovered models are identified and shown. Thereafter, a discussion is given of the propagation of the models in the CFD solver OpenFOAM. Furthermore, the effect of normalising the input library is examined, as well as the use of constraints and training on multiple cases. Finally, the results of including a damping function in the modeling framework are analysed.

6.1. Discovered Models

For each case described in Chapter 5, models are discovered with the sparse symbolic regression method as described in Chapter 4 using LASSO, Enet, SR3 or STLSQ regression and two different libraries of candidate functions. The first library only contains the invariants I_1 and I_2 from Table 4.2, including higher order functions of the invariants. The second library contains the invariants and additional physical features from Table 4.3. Each regression problem is solved for a range of hyperparameters to discover multiple model structures. Recap that the used hyperparameters for each regression problem are as follows

- $\lambda_{LASSO} \in [10 \cdot 10^{-4}, 9.5 \cdot 10^{-4}, 9 \cdot 10^{-4}, 8.5 \cdot 10^{-4}, ..., 8.5, 9, 9.5, 10]$
- $\lambda_{Enet} = \lambda_{LASSO}$
  $\rho_{Enet} \in [0.01, 0.1, 0.2, 0.5, 0.7, 0.9, 0.95, 0.99, 1.0]$
- $\lambda_{STLSQ} \in [0.001, 0.05, 0.1, 0.5]$
  $theshold_{STLSQ} \in [10 \cdot 10^{-3}, 9.5 \cdot 10^{-3}, 9 \cdot 10^{-3}, 8.5 \cdot 10^{-3}, ..., 8.5, 9, 9.5, 10]$
- $threshold_{SR3} = theshold_{STLSQ}$
  $\nu_{SR3} \in [0.01, 0.1, 1, 10]$

Thus, each regression problem is solved for every value or combination of values in the sets of hyperparameter. The values of the parameters influence the amount of non-zero coefficients and hence the level of complexity of the models. Furthermore, models are discovered for three regression settings. In the first setting, the libraries are not normalised before solving each regression problem. The outcome of the regression problem can directly be used. In the second setting, the libraries are normalised before solving each regression problem. Thereafter, an additional Ridge regression is performed on the not normalised selected functions to obtain a model having the correct units. For the additional Ridge regression we only have used one value for the hyperparameter, $\lambda = 0.1$. In the last setting, constraints are added to the regression problems and the libraries are not normalised. To obtain a simple model, the discovered models with five or less non-zero coefficients are selected and propagated. When the discovered models are propagated in OpenFOAM as described in Section 4.5.2, it can happen that the simulation does not converge. In that case the residuals in the simulation do not decrease. When the residuals decrease we say the simulation converged. Furthermore, a stopping criteria of 10^{-8} is applied.



(a) LASSO **(b)** Enet **(c)** SR3 **(d)** STLSQ

Figure 6.1: The models with five or less terms, discovered by solving one of the regression problems. The coloured blocks in the left graph of the figures indicate the coefficient values of selected functions. The blocks in the middle graph (green for yes and red for no) indicate if the model is realisable after training (first column) and after the implementation in a CFD solver (second column) and if the implemented model converged (third column). The mean squared errors ϵ of the predicted anisotropic stress and Reynolds stress as obtained during training and the error of the streamwise velocity as obtained after model propagation, normalised by the mean squared errors of the $k-\omega$ RANS model are plotted in the right graph.

Figure 6.1 shows for each regressor the discovered models containing five or less terms, trained on the PH case using the not normalised library with physical features. For example, the LASSO regression problem discovered the model $b_{ij}^{\Delta} = 0.32733T_1 - 0.38515T_1q_3 - 3.93275T_2$. In this case the value of the hyperparameter λ was $1.0 \cdot 10^{-4}$. Furthermore, the errors are shown of the predicted anisotropic and Reynolds stresses during the model discovery phase (before propagation) and of the corrected streamwise velocity as obtained when the model is propagated. In order to predict the stresses before propagation, RANS data is inserted into the discovered model to obtain $\hat{b}_{ij}^{\Delta}$ and the stresses are calculated by

$$\begin{aligned} \hat{b}_{ij} &= b_{ij}^{RANS} + \hat{b}_{ij}^{\Delta}, \\ \hat{R}_{ij} &= 2k^{RANS}(b_{ij}^{RANS} + \hat{b}_{ij}^{\Delta} + \frac{1}{3}\delta_{ij}). \end{aligned} \tag{6.1}$$

As the error metric the mean scared error (MSE) between the DNS data, X^{DNS} and predicted data $\hat{X}$ is used, which is defined as

$$\epsilon_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(X_i^{DNS} - \hat{X}_i)^2, \tag{6.2}$$

Here, n indicates the amount of data points. The same errors are calculated between the DNS data and the $k-\omega$ RANS predictions. The errors of the modeled variables are divided by the errors of the RANS predictions, such that it is easily observed if the model predicts better than the baseline $k-\omega$ RANS model. In Figure 6.1, it is also indicated if the simulation converged when the model was propagated on the same case, and if the modeled Reynolds stress was realisable before and after the propagation.

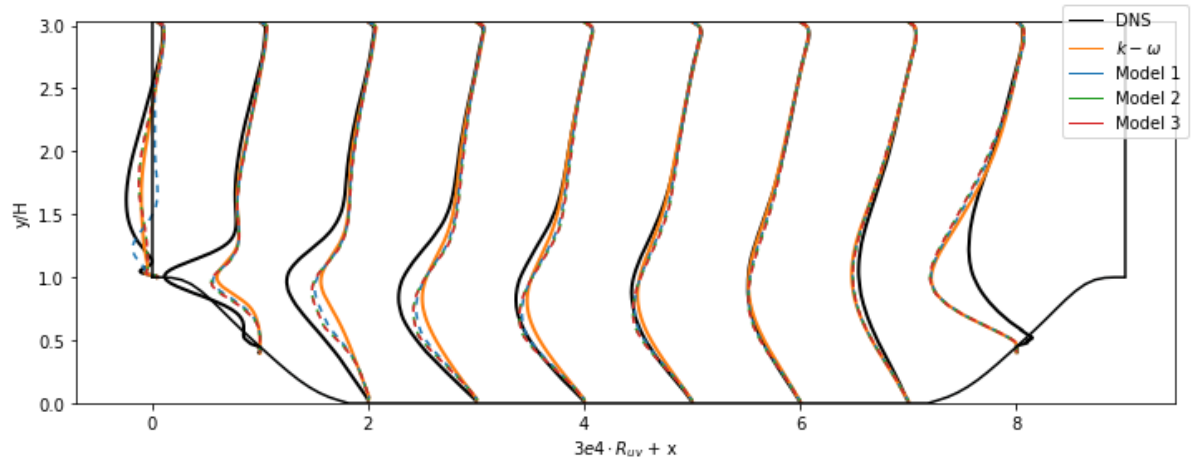
Table 6.1: Best performing models for the PH, CD and CBFS case discovered using the not normalised physical library and the normalised mean squared errors of the predicted streamwise velocity for each case.

Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = 70.3I_2T_2 + (-1177.3I_1 - 1248.8I_2 - 226.5q_1)T_3$	0.5639	0.8002	0.9744
$\mathcal{M}_2 = 96.3I_2T_2 + (52.4I_1 - 136.6I_2 - 29.5q_4)T_3$	1.1470	0.7271	0.9137
$\mathcal{M}_3 = 96.3I_2T_2 + (-84.3I_1 - 154.4I_2)T_3$	1.0924	0.7274	0.9124

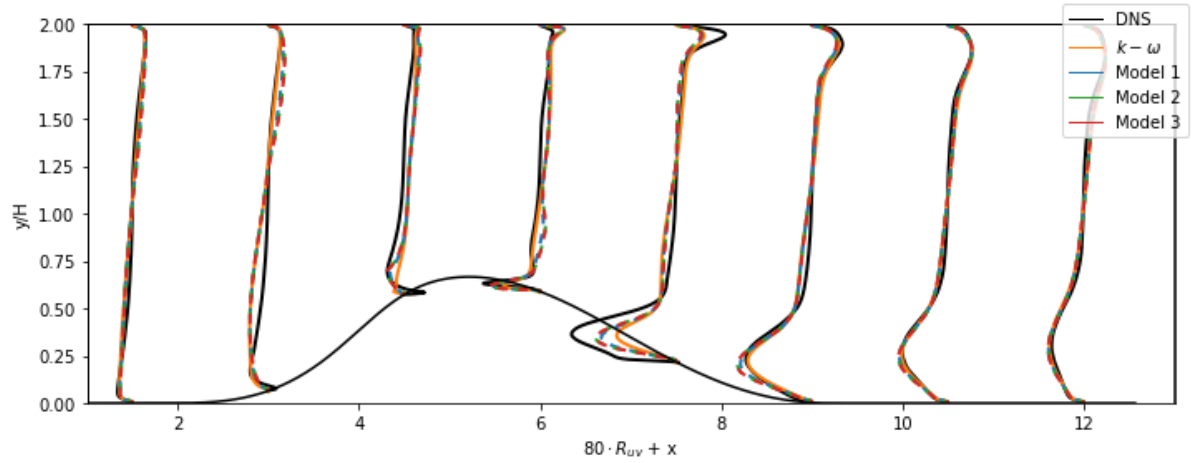
The realisability of the model is checked by analysing the Lumley triangle, introduced in Section 2.4.1. The Lumley triangle is created for the modeled Reynolds stress as obtained with equation 6.1 and for the Reynolds stress as outcome of the propagation. The discovered models found for the other cases, regression settings and library can be found in Appendices C and D.

Cross validation is carried out in order to identify the models with the best predictive capacities. This implies that models identified on one case are propagated on the remaining two cases. Thereafter, for each case, the best performing model is selected, which is the one having the lowest error for the predicted streamwise velocity. The best models are listed in Table 6.1, in which the corresponding streamwise velocity errors of the three cases are noted as well. These models are all discovered with the non-normalised library containing additional physical features. The best models discovered with normalised library and when constraints are added can be found in Appendix C, as well as the best models discovered with the invariant library in Appendix D. Comparing the best models of the two libraries, we can presume that using additional physical features results in better predicting models for the velocity than when only invariants are used. Furthermore, the best models which are found with no normalisation have lower predicting errors than models which are found with normalisation.

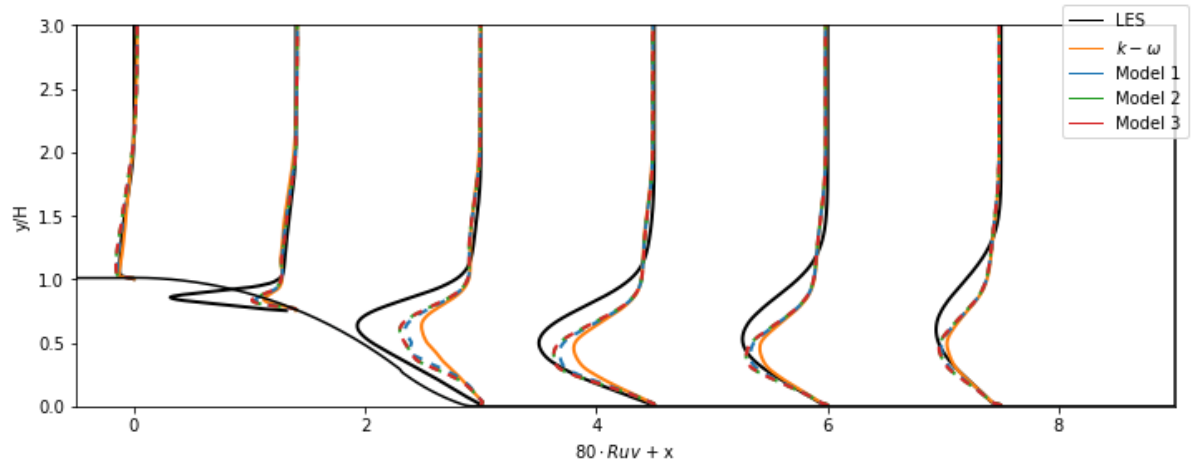
The models in Table 6.1 predict the shear Reynolds stress R_{uv} on each case as shown in Figure 6.2. The predictions of the Reynolds stresses are obtained with equation 6.1, before propagating the models in the CFD solver. The models show an improvement of the Reynolds stress on the descending slopes in all the cases in comparison to the baseline prediction of the $k-\omega$ model. Further away from these slopes the models are similar to the prediction of the $k-\omega$ model. Propagating the models in OpenFOAM gives us a prediction of the velocity and the kinetic energy. The results for the streamwise velocity and the kinetic energy can be seen in Figures 6.3 and 6.4, respectively. For the PH case, the velocity prediction of model 1 is well captured and improves the prediction compared to the $k-\omega$ model throughout the whole domain. Model 2 and 3 fit the data only better for $x > 6$. On the CD case the models show improvements of the velocity predictions down the slope. For the CBFS case the models show the smallest improvements. For all the cases, the kinetic energy is generally better captured by each model than the $k-\omega$ baseline, but the improvements are limited.



(a) PH

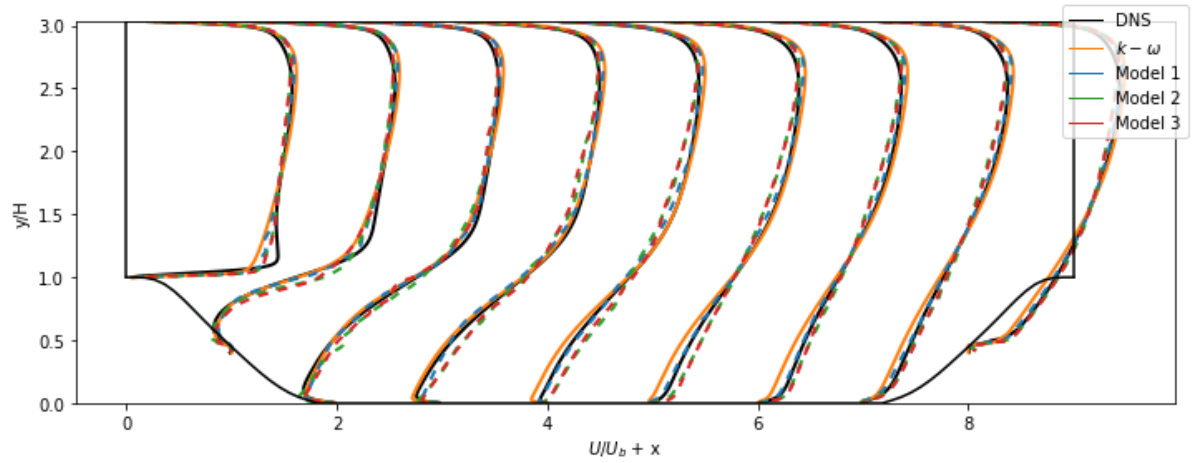


(b) CD

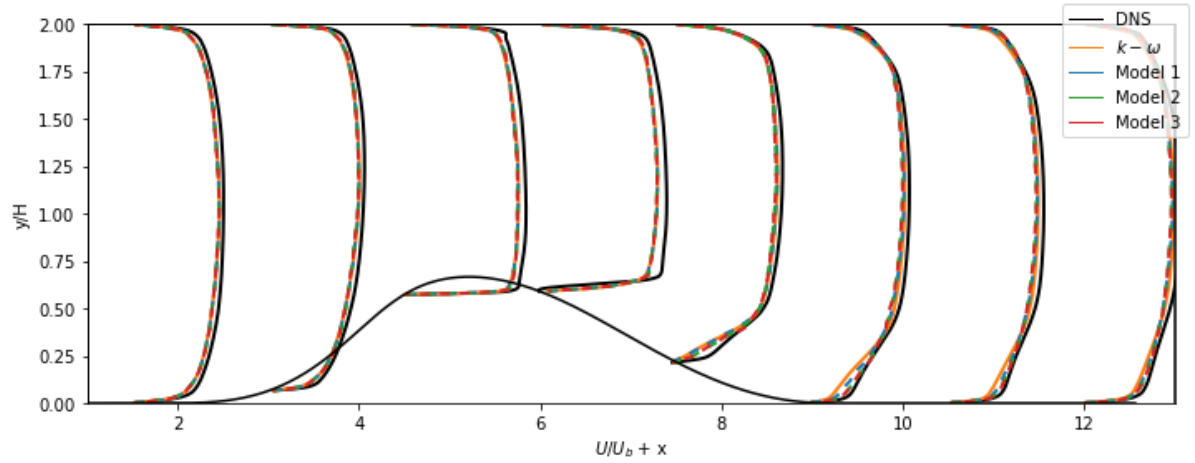


(c) CBFS

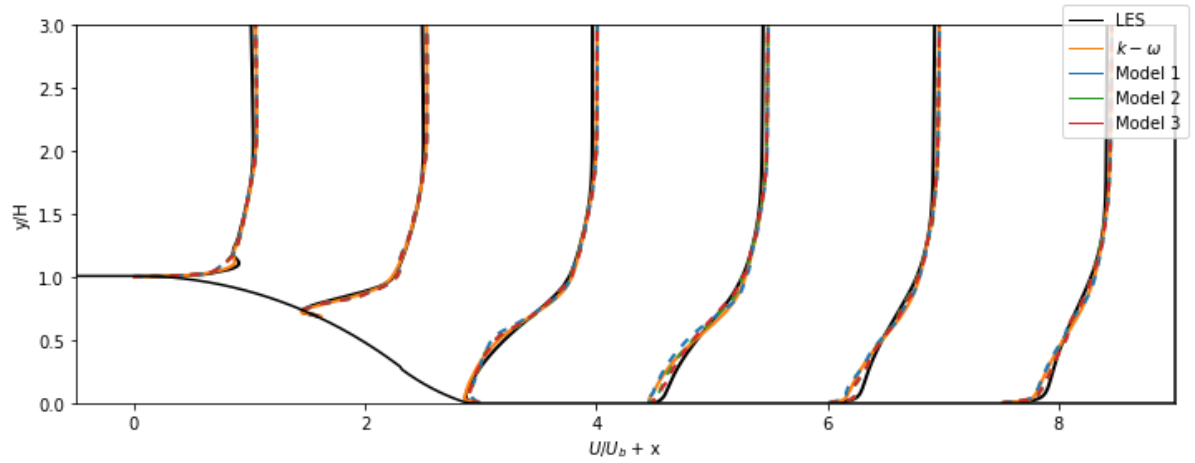
Figure 6.2: Reynolds shear stress predictions of the best performing models (Table 6.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

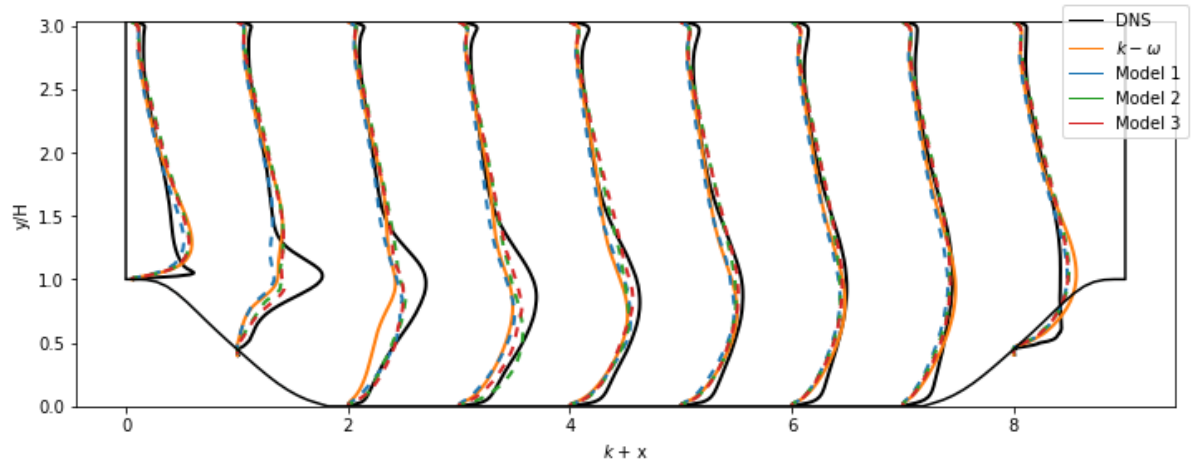


(b) CD

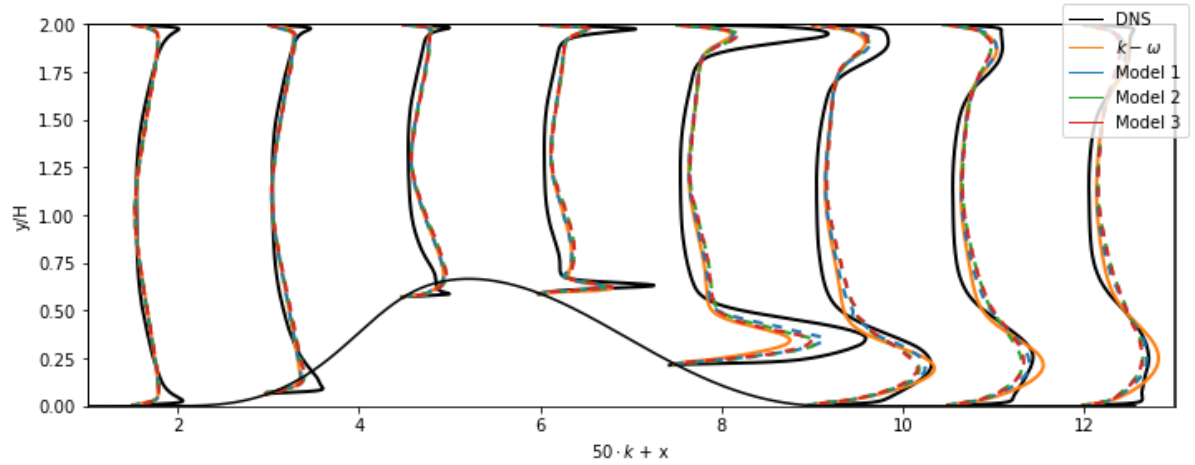


(c) CBFS

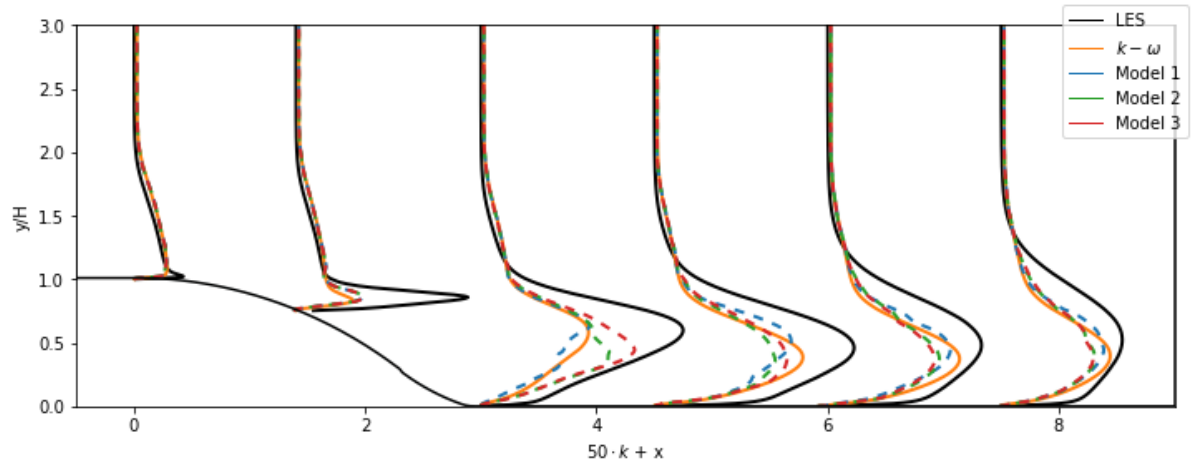
Figure 6.3: Streamwise velocity predictions of the propagated best performing models (Table 6.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH



(b) CD



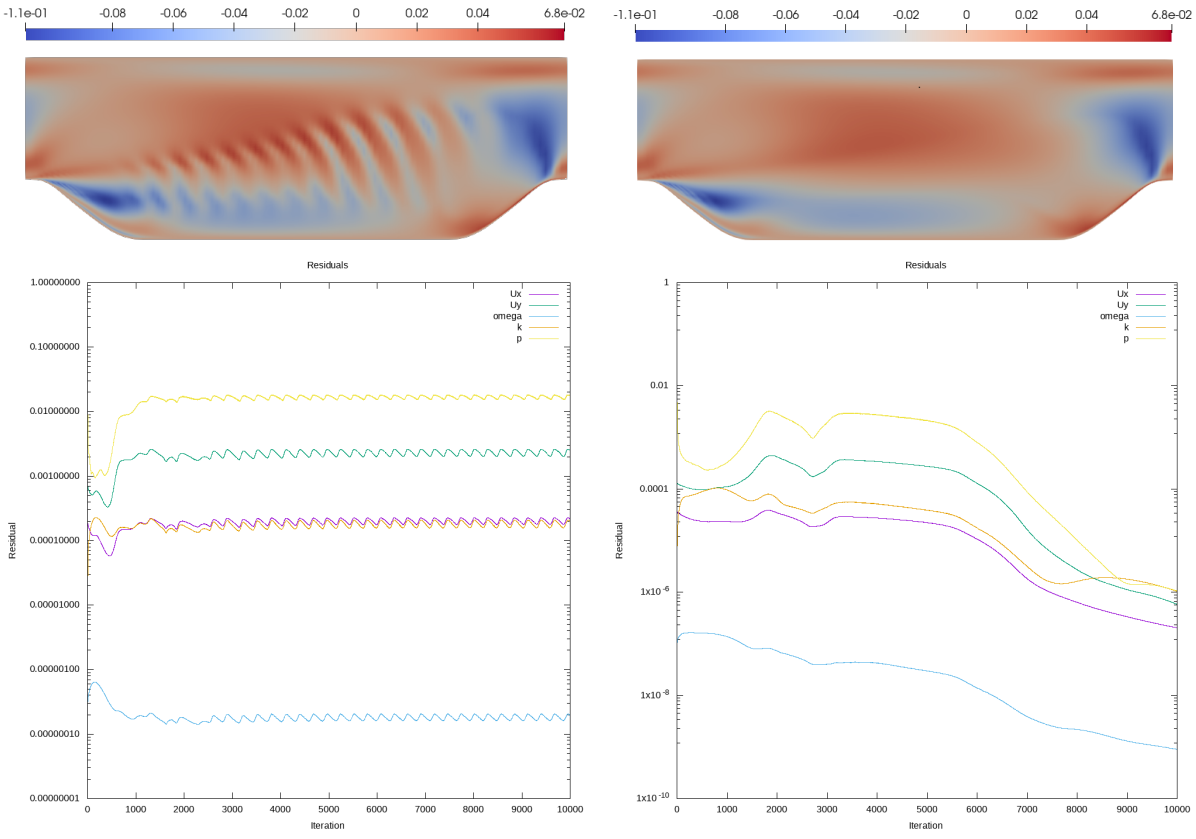
(c) CBFS

Figure 6.4: Kinetic energy predictions of the propagated best performing models (Table 6.1) compared to DNS and $k-\omega$ RANS data for each case.

6.1.1. Propagating Instabilities

Figures 6.3 and 6.4 may suggest that the discovered models improve the $k-\omega$ RANS model, but the results of the propagation for the anisotropic and Reynolds stresses imply differently. The predictions of the stresses show unexpected deficiencies. Figure 6.5a shows an example of such deficiencies. These kind of instabilities in the propagation appear for the majority of the discovered models. In all the cases where instabilities are visible, the simulations did not converge. One cause of the appearance of instabilities could be that the addition of the correction term results in relatively high or unfeasible values and gradients at some points in the domain, such as the separation point. High values at one point can lead to numerical difficulties affecting the rest of the domain. This assumption is supported by the observation that lowering the relaxation factor for U in OpenFOAM eliminates the instabilities in some cases, which is explained in Section 4.5.1. The relaxation factor controls the amount which a variable changes from one iteration to the next. As an example, improving the stability by lowering the relaxation factor is seen for the model $\mathcal{M} = -4.32T_{ij}^{(2)}$. The residual plot of the simulation with the same relaxation factor 0.7 for U, k, ω and p can be seen in Figure 6.5a together with the contour plot of b_{uv}^{Δ}. This plot is a typical example of how the instabilities look like, but for different propagated models more unstructured patterns are observed as well. When we lower the relaxation factor for U to 0.3, the simulation converges, as can be seen in the residual plot in Figure 6.5b. Simultaneously, the deficiencies in the flow disappear. However, this is only observed for a few models on the PH case and not for the CD and CBFS cases.

Furthermore, models containing the same functions can perform differently in terms of stability. For instance, the model $\mathcal{M} = 0.59T_{ij}^{(1)} - 1.71T_{ij}^{(2)}$, discovered by the LASSO regressor on the CBFS case, converges on all three test cases. The model $\mathcal{M} = 0.46T_{ij}^{(1)} - 3.38T_{ij}^{(2)}$, discovered on the CD case, only converges for the CBFS case, but does not converge on the PH and CD case. This indicates that



(a) Residuals and contour plot b_{uv}^{Δ} of the simulation using relaxation factors 0.7 for U, k, omega and p.

(b) Residuals and contour plot b_{uv}^{Δ} of the simulation using relaxation factors 0.3 for U and 0.9 for k, omega and p.

Figure 6.5: Lowering the relaxation factors as used in the model propagation improves stability.

the outcome of a model can be crucial for the stability of the simulation. It is yet possible that some functions are more likely to give this unstable outcome. It was observed that more stable models are discovered after removing specific features (e.g. feature q_7) from the library, that are often present in unstable models.

Another cause of the instabilities could be that the added models predict unrealisable Reynolds stresses or give unfeasible values for one of the flow variables. To check this, for each model, the Lumley triangles are examined before and after the implementation of the model in OpenFOAM. However, a strong correlation between realisability and convergence is not discovered. There are both converging models with non-realisable Reynolds stresses and non-converging models with realisable Reynolds stresses. Furthermore, if the stresses are realisable before propagation, it is not guaranteed that they are still realisable after propagation and that the model converges. Even more surprisingly, Reynolds stresses can be unrealisable before the propagation, and realisable after propagation. However, it is still possible that unfeasible values in the simulation can cause the instabilities.

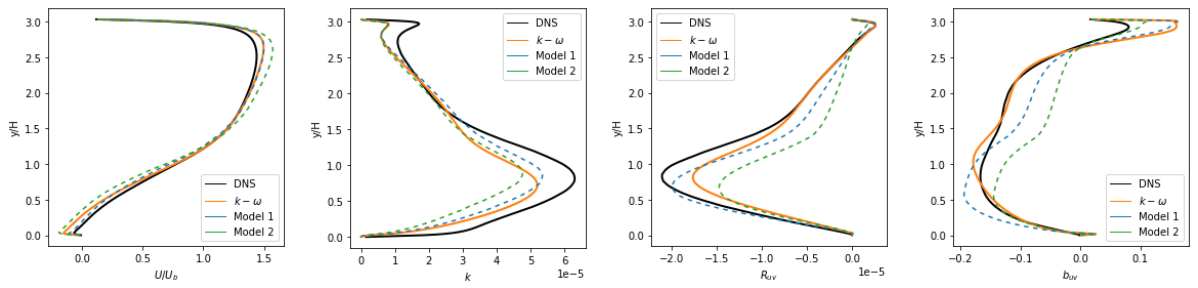
6.1.2. Model Analyses

Taking a closer look at the results of all the models, it is interesting to note that not all unconverged models lead to bad results. There are unconverged results which are able to predict the streamwise velocity rather well, as seen for the best models in Table 6.1. Simultaneously, there are converged models which predict the streamwise velocity poorly. One of the advantage of sparse symbolic regression is its open-box methodology. For each model, it is exactly known which variables are present. Hence, we observe that models containing tensor $T_{ij}^{(1)}$ always give a poor prediction of the velocity profile. However, these models are not always unconverged. Furthermore, models containing $T_{ij}^{(1)}$ have poor predicting results for the kinetic energy, which are even lower than the under-prediction of the $k-\omega$ model. Figure 6.6 compares two models obtained by solving the LASSO regression problem using the normalised library with only invariants defined by

$$\mathcal{M}_1 = -4.32T_2, \tag{6.3}$$

$$\mathcal{M}_2 = 0.325T_1 - 4.291T_2. \tag{6.4}$$

The first model does not contain $T_{ij}^{(1)}$, but the second model does. The predictions of the velocity and kinetic energy of both models on the PH case can be seen in Figures 6.6a and 6.6b. In these figures, the quantities are plotted on one cross-section of the channel, $x = 4$. The different performance of the two models could emanate from a different modeling of the uv-component of the anisotropic stress. As can be seen in Figure 6.6d, model 2 predicts the anisotropic stress near the upper wall better than model 1 and the $k-\omega$ model but under-predicts the stress in the rest of the domain. This under-prediction is also visible for the shear Reynolds stress in Figure 6.6c. The diagonal components of the anisotropic and Reynolds stress of the two models are almost identical. This suggests that $T_{ij}^{(1)}$ mainly affects the uv-components of the Reynolds stress. Furthermore, an under-prediction of the shear Reynolds stress in the inner part of the domain can be related to an under-prediction of the kinetic energy and a poor



(a) U **(b)** k **(c)** R_{uv} **(d)** b_{uv}

Figure 6.6: Influence of tensor $T_{ij}^{(1)}$ on the prediction of the stresses, kinetic energy and velocity, inspected by comparing the models defined by equations 6.3 and 6.4.

prediction of the streamwise velocity. To improve this, the modeling framework could be expanded such that corrective models of the kinetic energy are discovered as well, as done by Schmelzer *et al.* [15]. These models could also provide more stability to the propagation of the models.

6.2. Normalisation

Normalisation of the library has an effect on the function selection, the model performance and the model generalisability. When considering only the library with additional physical features, normalising the library results in better predictions of b_{ij} and R_{ij} before model propagation, see Figure 6.7a. That is, the models obtained with normalisation have less outliers upward than the models obtained with the non-normalised library. In Figure 6.7a, each marker represents a discovered model on the PH case by solving one of the regression problems. A filled marker indicates a converged model, where an empty marker indicates that the model did not converge. When models are discovered with the normalised library an additional regression step is done to obtain a model containing the original non-normalised functions as explained in Section 4.2.5. This additional regression step can also be seen as an extra calibration of the selected functions, which is likely to give better models for the training data. If we look at the propagating results for U in Figure 6.7a, the models obtained without normalisation perform equally or better than models found with normalisation and have less outliers upward. Furthermore, the best predictions of U are of models discovered using the non-normalised library and hence without performing an additional Ridge regression. This is observed for the other cases as well, of which the results can be found in Appendices C.5 and D.5.

A good model should be capable to predict the Reynolds stress and the velocity for multiple cases. Figures 6.7b and 6.7c show the results of models trained on the PH case and tested on the CD and CBFS cases. Especially for the CD case, the best predictions for the Reynolds stress of the models obtained without normalised functions are better than of the models obtained with normalised functions. The best predictions of the velocity are also slightly better. On the CBFS, a very small improvement is observed for the non-normalised functions. This suggest that not normalising the features (and not performing the additional Ridge regression) results in more generalisable models. However, for the models trained on the CD and CBFS case we see less improvements of the model performances on the other cases when the library is not normalised. In general, the predicting errors of models trained on one case are equally spread across the different cases which they are tested on. This suggest that models perform similar on the different cases.

However, a direct comparison of the performance of models obtained with and without normalised libraries is tricky because of two reasons. First, the additional Ridge regression performed when the

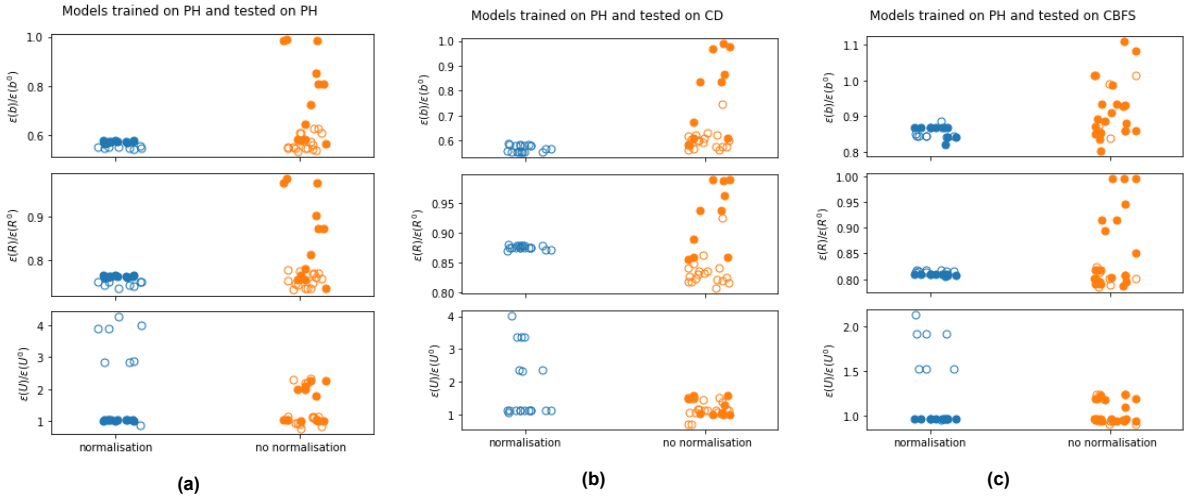


Figure 6.7: Comparison of MSE of predicted b_{ij}, R_{ij} and U of all models discovered on the PH case using a normalised or not normalised library and tested on the PH, CD and CBFS. A filled or empty marker represent that the model converged or not during propagation, respectively.

library is normalised can have much influence on the performance of the model. Therefore, it is unfair to compare these models with the models obtained without normalising the library and not performing an additional Ridge regression. Eliminating the additional Ridge regression would provide a better comparison. This could be done by rescaling the coefficients for the normalised functions in order to obtain coefficients for not normalised functions. Secondly, different functions are selected when the library is normalised compared to when it is not normalised. Figure 6.8 shows for each regression problem the selected functions. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function. While the poorly predicting models always contain $T_{ij}^{(1)}$, the combinations of $T_{ij}^{(1)}$ with the input features are different. For instance, solving the LASSO problems results in models with $q_7 T_{ij}^{(1)}$ and $q_1 T_{ij}^{(1)}$ when the library is normalised, but discovers models with $q_3 T_{ij}^{(1)}$ when the library is not normalised. Therefore, the differences in performance between models obtained with and without normalised libraries are likely caused by a different selection of the functions, rather than the normalisation itself.

Normalising the library has a different effect for each regression problem. The observation that normalisation improves the predictions of b_{ij} and R_{ij} but not the propagating performance of U mainly holds for the LASSO and Enet regression problems, as can be seen in Figures 6.9a and 6.9b. Additionally, small differences are observed between the function selection of the normalised and non-normalised library for the LASSO and Enet optimisers. For instance, when the functions are normalised Enet and LASSO select the invariants, but when they are not normalised they are never selected. This is because Enet and LASSO prefer models with low values for the coefficients. In the non-normalised library the invariants have lower values compared to the physical features. Therefore, they require higher coefficient values to compete with the physical features, hence unlikely to be selected by Enet or LASSO. Compared to the SR3 and STLSQ regression problems, models discovered by Enet or LASSO perform worse. However, this is mainly because these models often contain tensor $T_{ij}^{(1)}$, while models discovered by SR3 and STLSQ are less likely to contain this tensor. Furthermore, we observed that the model selection of LASSO and Enet problems depends on the initial guess in the used solving algorithm, which is the coordinate descent algorithm. When the library has a different order of the candidate functions, different functions are selected. This can be the result of the coordinate descent method finding a different local minimum for a different initial guess, which is used for the LASSO and Enet problems.

The models discovered by the SR3 and STLSQ problems perform similarly when the normalised library is used as when the not normalised library is used. Both regression problems are based on Ridge regression. Hence, the additional Ridge regression as done after solving the SR3 and STLSQ problems for normalised libraries does not change the models considerably. Normalising the library mainly

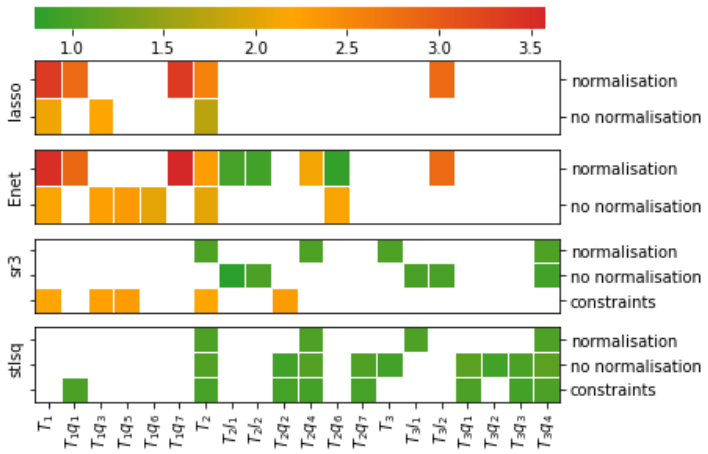
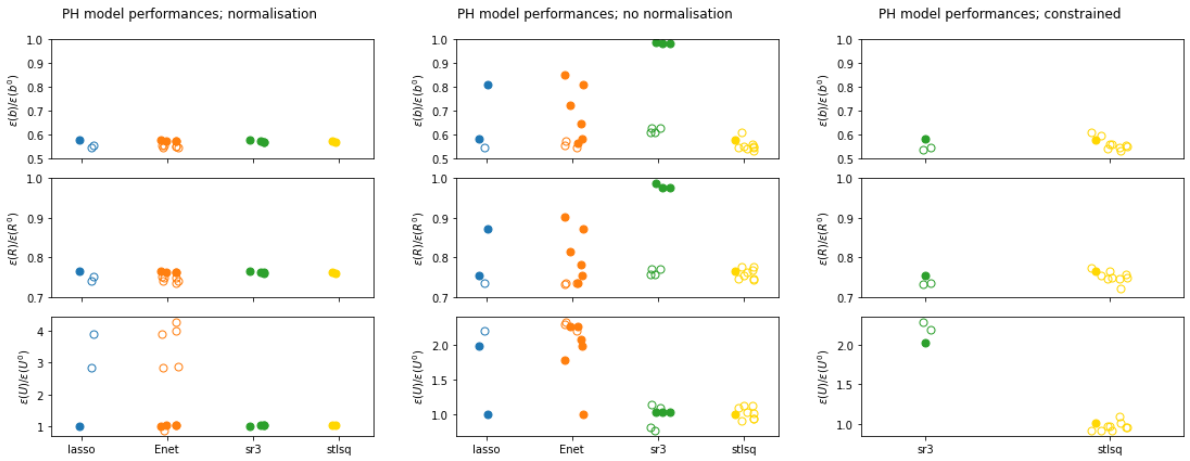


Figure 6.8: Feature selection per regression problem and setting for the PH case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.



(a) Models trained with normalisation **(b)** Models trained without normalisation **(c)** Models trained with constraints

Figure 6.9: Comparison of the MSE of predicted b_{ij}, R_{ij} and U of all models discovered and tested on the PH by one of the four regression problems. A filled or empty marker represents that the model converged or not during propagation, respectively.

has effect on the amount of models which are discovered by the SR3 and STLSQ regressors. When the library is not normalised, the STLSQ optimiser discovers more diverse models and selects more and different functions. That is, different values for the hyperparameters result in different models. This makes sense, since STLSQ selects functions based on the value of the coefficient. When the functions are normalised, the range of the coefficient value will be small and there will be little threshold values that produce different models. When the functions are not normalised, the range of the coefficients is larger and a different cut-off threshold is more likely to result in different models. The SR3 optimiser selects totally different functions when they are not normalised compared to when they are, for a similar reason as the STLSQ optimiser. SR3 as used in this research performs hard thresholding on the value of the coefficients. Since the non-normalised invariants have a lower value than the physical features, the regularised coefficients of the invariants will be larger and therefore are unlikely to be cut-offed.

So far, only the models are discussed which are found with the physical library. For the invariant library we observe similar patterns in terms of function selection. First of all, the LASSO and Enet regressors only select higher order functions when the library is normalised. The higher order functions, such as $I_1^2 T_2$, have lower values than functions like $I_1 T_2$. Hence, they require higher coefficient values when not normalised and are less likely to be selected by these regressors. The STLSQ problem selects much more different models when the library is not normalised for the same reason as we saw for the physical library. The SR3 problem selects more higher order functions when they are not normalised compared to when they are. However, the coefficient values of these functions are very high, which resulted in problems during the model propagation.

Furthermore, the differences between the models obtained with and without normalising the invariant library are similar as seen for the physical library. The predictions of b_{ij} and R_{ij} contain less outliers when the library is normalised. However, the best predictions are of the models which are obtained without normalising the library. The predictions for U of models using the normalised library are mostly comparable to models using the not normalised library.

From the results given in this section, we may conclude that normalising the library mainly leads to a different selection of the candidate functions. The preference of normalising the library or not is dependent on the used regression method and the values of candidate functions. In our case, non-normalisation of the library is preferred for the SR3 and STLSQ regressors, since that results in a wider variety of models discovered by these regression methods. For the LASSO and Enet regression problems, normalising the library is favoured to include functions with relatively low values as well. A direct comparison between the performances of models obtained with and without normalising the library is difficult since normalised libraries underwent an additional Ridge regression and non-normalised

libraries not. However, the best performing models are obtained when the libraries are not normalised.

6.3. Constraints

Constraints are applied to force realisability of the trained models. In order to show the effectiveness of the added constraints, we compare models obtained with and without constraints, both using the non-normalised library of physical features. The unconstrained STLSQ optimiser discovered the model

$$b_{ij}^{\Delta} = (-47.0q_1 + 6.2q_6 + 9.6q_7)T_3. \tag{6.5}$$

The Reynolds stresses predicted by the model before propagation are not realisable, illustrated with the Lumley triangle in Figure 6.10a. When constraints are added, all the models discovered by the constrained STLSQ regression problem predict Reynolds stresses which are realisable. One of the discovered models is

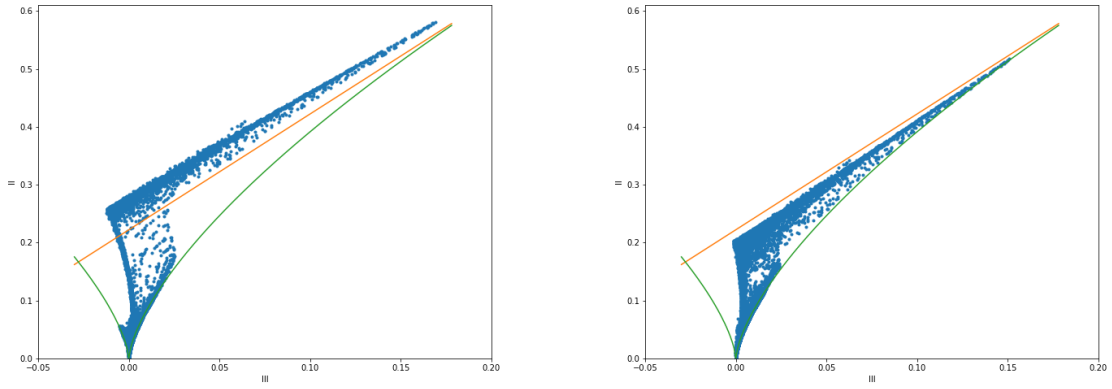
$$b_{ij}^{\Delta} = (-17.8q_1 + 9.9q_6 + 18.1q_7)T_3. \tag{6.6}$$

The Lumley triangle of this model can be seen in Figure 6.10b. The constrained STLSQ optimiser has selected the same functions, but with different coefficient values.

In order for the symmetric tensor b_{ij} to be in the desired intervals, six constraints are required as introduced in equation 4.13. However, the implementation demanded to rewrite the six constraints into twelve constraints of the form $b_{ij} \leq a$, compiled into one vector. For a case with n points and a library of p functions, the constraints compose an additional matrix of $12n \times p$. For $n = 15.000$ as used in our cases, this led to memory issues for some of the regression problems. Since the cases are statistically two dimensional, we imposed that only b_{uu}, b_{uv} and b_{vv} have to be in the interval, which requires only six constraints of the form $b_{ij} \leq a$ and ease the memory capacity.

It was not always possible to find models containing less than five terms which satisfy the added constraints. We experienced that the threshold value used in the STLSQ optimiser could be too high to solve the problem. Consequently, the threshold was lowered such that many non-zero coefficients remain. Especially for the CD case, it was hard to find simple models satisfying the constraints, possibly because the $k - \omega$ RANS simulation of the CD case was not realisable as well. Therefore, the Reynolds stresses of almost every discovered model were not realisable.

Considering only the library with additional physical features, adding constraints to the STLSQ problem does not lead to a big difference in function selection; see Figure 6.8. It is mainly the value of the coefficients which changes, as seen in equations 6.5 and 6.6. Adding constraints to the SR3 problem does lead to a different selection of features. None of the functions which are selected without additional constraints are selected when constraints are added. This is possibly due to the fact that the



(a) Without constraints

(b) With constraints

Figure 6.10: Lumley triangles of the models discovered with additional constraints to the STLSQ regression problem 6.6 and without 6.5.

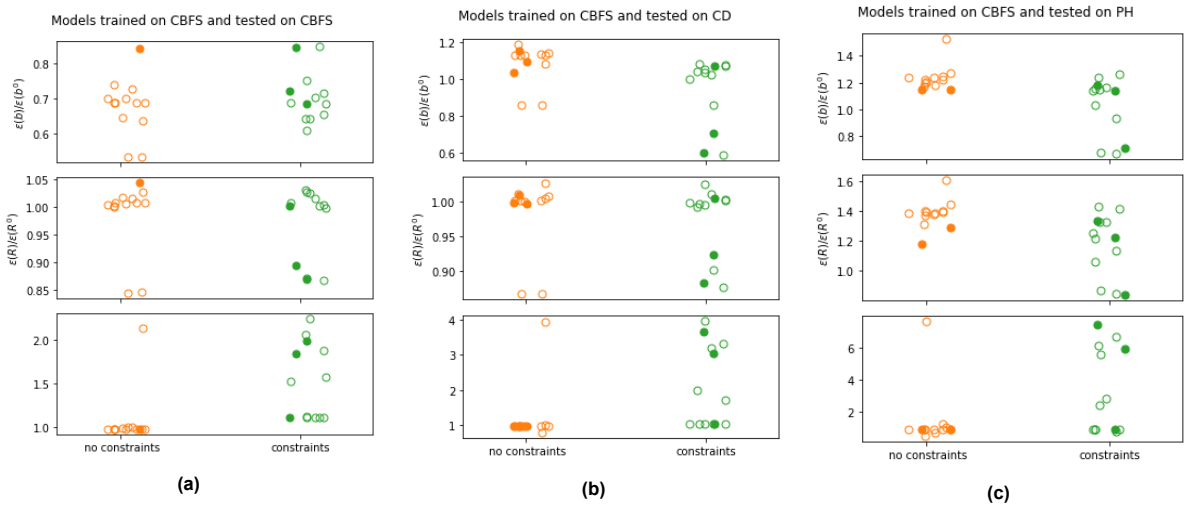


Figure 6.11: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the CBFS case with and without additional constraints, tested on the CBFS, CD and PH case. A filled or empty marker represent that the model converged or not during propagation, respectively.

constrained SR3 method is implemented with a different norm of the penalty term than the SR3 method without constraints. The constrained SR3 method contains the l_1-norm, while the SR3 regression problem without constraints performs hard thresholding.

The differences in the function selection between the constrained and unconstrained regression problems can explain the differences in performance. For the STLSQ problem, which selects similar functions with and without additional constraints, we see that the performances of the discovered models when constraints are added are also similar compared to the models discovered without constraints, see Figure 6.9c. The mean squared errors of predicted b_{ij}, R_{ij} and U are equally spread. However, for the models discovered with the SR3 problem the propagating results for U are worse when constraints are added. This is likely caused by the appearance of $T_{ij}^{(1)}$ terms in the models discovered when constraints are added. This results in different performances between the models discovered with SR3 and STLSQ. The models discovered with constrained STLSQ have generally better propagating results for U with less outliers than the models discovered with the constrained SR3 problem.

In general, adding constraints does not improve the performance of models on the same case as which they are trained on; see Figure 6.11a. However, the performance on other cases can improve when constraints are added. This suggest that constraints stimulate the generalisability. Figure 6.11b and 6.11c show that models trained on the CBFS with constraints and propagated on CD and PH have lower errors of predicted b_{ij} and R_{ij} than models without constraints. Unfortunately, this is less clear in the propagating results for U. Finally, we hoped to see an improved stability of the models obtained with additional constraints, but this is not observed. Only a small amount of the discovered models converged when propagated, which is indicated by a filled marker in Figure 6.11.

6.4. Training Multiple Cases

Besides discovering models on each case individually, models are trained on a combined dataset of the three cases. Of the CD and CBFS cases, 15.000 points are randomly selected such that an equal amount of data is used of each case. It is observed that models which are trained on multiple cases generally have lower predicting errors of b_{ij} and/or R_{ij} than the models which are trained on a single different case. Figure 6.12 shows the errors of models trained on all cases or a single case using the normalised physical library and tested on the CBFS. We can see that the errors for b_{ij} are lower for the models trained on all cases than when trained on one different case. However, the errors for R_{ij} are more similar. Training on all cases does not improve the prediction of U or the amount of converged models. This was seen for both libraries and all the cases, of which the results can be found in Ap-

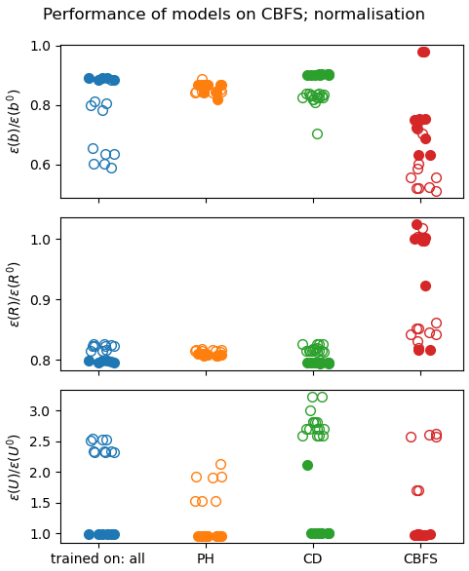


Figure 6.12: Comparison of models trained on all cases and trained on a single case, tested on the CBFS case.

pendices C.5 and D.5. The results of models obtained with normalised libraries are similar to models found without normalising the libraries. We were not able to obtain simple models for the constrained regression problems when trained on the mixed data set. The bottleneck is probably the data of the CD case, since solving the constrained problems of this single case did not generate simple models as well. However, this is not further tested nor confirmed.

6.5. Damping

As closure of this chapter, the results are discussed of introducing near-wall damping to the modeling framework. As explained in Section 4.4, two methods are developed to include the van Driest damping function. In the first method, damping is added manually after model discovery and in the second method, damping functions are added to the library. These two methods will be referred to as the 'manual' method and the 'library' method, respectively. To obtain a better understanding of the near-wall behaviour, initially a simple fully developed turbulent channel flow is considered. For this case, the DNS data of a fully developed plane turbulent channel flow with $Re_\tau = 392.24$, generated by Moser, Kim and Mansour is used as reference data [75]. The same flow case is simulated with the $k-\omega$ RANS model in OpenFOAM. To obtain a flow with the same Re_τ, we used the physical parameters as stated in Table 6.2. Table 6.3 contains the numerical boundary conditions as used in the RANS simulation.

6.5.1. Channel Flow

Applying the manual method, the best model for the damped anisotropic stresses discovered by the constrained STLSQ regression problem using the invariant library is

$$\mathcal{M}_1 = -4.01T_2 + 3.71T_3. \tag{6.7}$$

Thereafter, the damping function is implemented following equation 4.21. Using the library method, the best discovered model for the same library but multiplied with the damping functions discovered by the

Table 6.2: Physical parameters as used in the $k-\omega$ RANS simulation of a fully developed turbulent plane channel flow.

U_τ	$(1.0, 0.0, 0.0)\ m/s$
ν	$0.002532\ m^2/s$
U_b	$(17.55, 0.0, 0.0)\ m/s$

Table 6.3: Boundary and initial conditions used in the $k-\omega$ RANS simulation of a fully developed turbulent plane channel flow.

	Inlet	Outlet	Walls	Sides	Internal field
U	Cyclic	Cyclic	No Slip	Empty	$17.55\ m/s$
p	Cyclic	Cyclic	zeroGradient	Empty	$0.0\ m^2/s^2$
k	Cyclic	Cyclic	kqRWallFunction	Empty	$0.17\ m^2/s^2$
ω	Cyclic	Cyclic	omegaWallFunction	Empty	$0.083\ s^{-1}$
ν_τ	Cyclic	Cyclic	nutUWallFunction	Empty	$0.623\ m^2/s$

constrained STLSQ problem is

$$\mathcal{M}_2 = (-7.5 - 0.44f + 4.35f^2)T_2 + (1.78 + 1.92f - 1.78f^2)T_3 \tag{6.8}$$

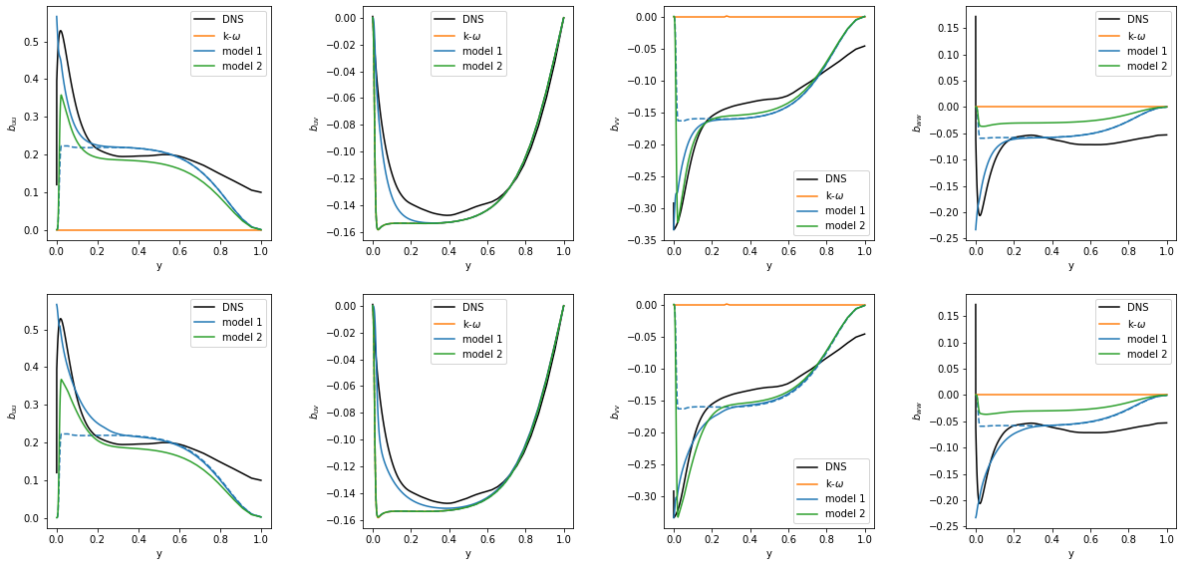
The model predictions of the $uu-$, $uv-$, $vv-$ and $ww-$components of the anisotropic Reynolds stress are presented in the top row of Figure 6.13. In these graphs, the full blue line represents the damped model obtained with the manual method. The dashed blue line is the base of this model without damping, thus only equation 6.7. The green line represents the model obtained with the library method. The figures show that model 1 is able to approximate the DNS data for each component. Furthermore, the asymptotic behaviour of model 1 near the wall is nicely visible. Model 2 approaches the DNS data of the anisotropic stresses for the components uu, vv and ww, but has more difficulties with b_{uv}. This is induced by the absence of $T_{ij}^{(1)}$ in the model, which is the only tensor having effect on the shear stress for this one dimensional problem. Model 2 also has a less accurate prediction of the near-wall behaviour than model 1, yet much better than the base of model 1. At $y = 0$, the outcome of model 2 is zero, which is caused by the zero value of the base tensors $T_{ij}^{(n)}$ at the wall. This zero value originates from the scaling of the strain rate tensor S_{ij} and rotation rate tensor Ω_{ij} with $\tau = 1/\omega$. A more appropriate expression for the timescale would be $\tau = \max(\frac{k}{\epsilon}, C_\tau\sqrt{\frac{\nu}{\epsilon}})$, which was introduced by Durbin for the $k-\epsilon$ models. However, this has not been further investigated in this research, since the required modification to be applicable in the $k-\omega$ model showed singularities for the channel flow. The results of the models when propagated in OpenFOAM are presented in the second row of the same Figure 6.13. After propagation, the results for the anisotropic stress are very similar as before.

The model predictions of the uu-, uv-, vv- and ww-components of the Reynolds stress before propagating the models in OpenFOAM are presented in the top row of Figure 6.14. Here, model 2 is able to approximate the DNS data of all the Reynolds stress components. Model 1 approaches the DNS data of the diagonal components, but is less accurate for the Reynolds shear stress. However, both models perform better than the $k-\omega$ RANS model, and especially for R_{vv} an improvement of near-wall behaviour is nicely visible. The second row of Figure 6.14 shows the Reynolds stresses of the propagation of the model. The predictions of model 2 are similar as before, but the Reynolds stresses predicted by model 1 have been significantly changed. Moreover, compared to the propagation results of its original basis, the Reynolds stresses of damped model 1 are considerably lower. The under prediction of model 1 is also seen in the predicted kinetic energy, shown in Figure 6.15a. Furthermore, the friction Reynolds number Re_τ and the friction velocity u_τ are decreased, noted in Table 6.4. The changes of these values suggest that the flows scaled differently in the simulation. This is related to how the problem is defined in the CFD solver. For a better comparison between the models, we rescale the Reynolds stresses with u_τ^2. The corrected results can be seen in the bottom row of Figure 6.14. The scaled Reynolds stresses of model 1 approaches the DNS data and are again similar to the prediction of its base model. Since the Reynolds stresses of model 2 and the undamped base of model 1 are similar to their scaled stresses, these models are less affected by a different scaling in the simulation. The different near-wall behaviour of b_{uv} of these models compared to model 1 could be the source of the different scaling behaviour.

The results demonstrate the potential of including damping functions in the modeling framework, but care must be taken for devious side effects.

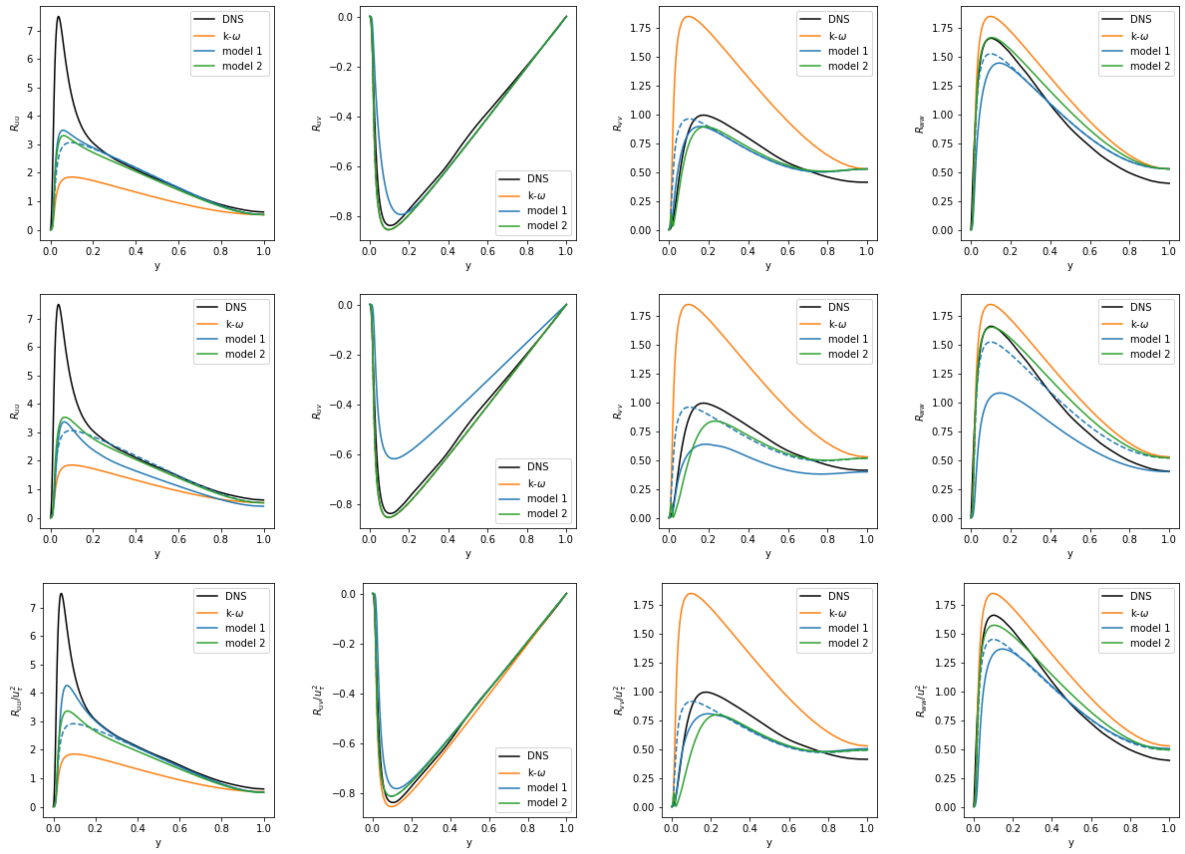
Table 6.4: The friction Reynolds number Re_τ and friction velocity u_τ of the $k-\omega$ RANS simulation and propagated models 1 and 2, defined by equations 6.7 and 6.8.

Model	u_τ	Re_τ
k-ω	1.0256	405
model 1 without damping	0.9863	390
model 1 with damping	0.8665	342
model 2	1.0248	405



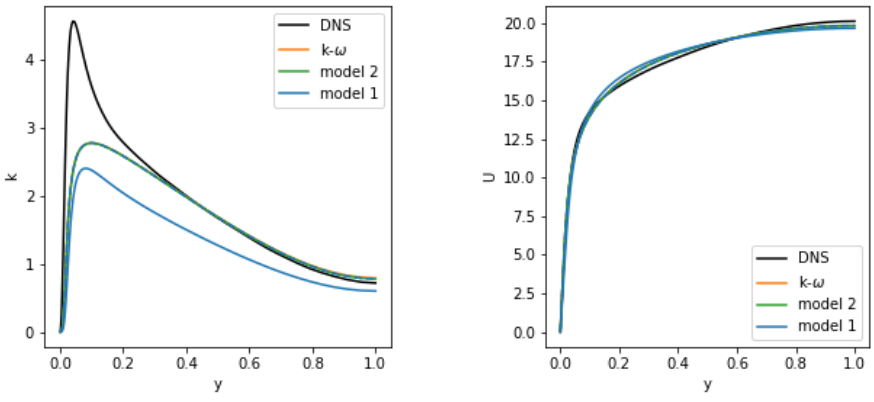
(a) b_{uu} **(b)** b_{uv} **(c)** b_{vv} **(d)** b_{ww}

Figure 6.13: Profiles of the uu-, uv-, vv- and ww-components of the anisotropic Reynolds stress in a turbulent channel flow as predicted by models 1 and 2 defined by equations 6.7 and 6.8, which include damping functions, compared to DNS data and the $k-\omega$ RANS model. Model 1 consists of a base model (blue dashed line) to which damping is added manually. The top and bottom rows represent the stresses obtained before and after propagating the models in OpenFOAM, respectively.



(a) R_{uu} **(b)** R_{uv} **(c)** R_{vv} **(d)** R_{ww}

Figure 6.14: Profiles of the uu-, uv-, vv- and ww-Reynolds stress components in a turbulent channel flow as predicted by models 1 and 2 defined by equations 6.7 and 6.8, which include damping functions, compared to DNS data and the $k-\omega$ RANS model. Model 1 consists of a base model (blue dashed line) to which damping is added manually. The first and second rows represent the stresses obtained before and after propagating the models in OpenFOAM, respectively. In the last row, the stresses obtained after propagation are scaled with u_τ^2.



(a) **(b)**

Figure 6.15: Profiles of the kinetic energy and streamwise velocity in a turbulent channel flow as predicted by models 1 and 2 defined by equations 6.7 and 6.8, which include damping functions, compared to DNS data and the $k-\omega$ RANS model. Model 1 consists of a base model (dashed line) to which damping is added manually.

6.5.2. Periodic Hill

The use of damping functions was also tested on the PH case. Applying the manual method, the best model for the damped anisotropic Reynolds stress, which does not contain tensor $T_{ij}^{(1)}$, discovered with the constrained STLSQ regressor and the invariant library is

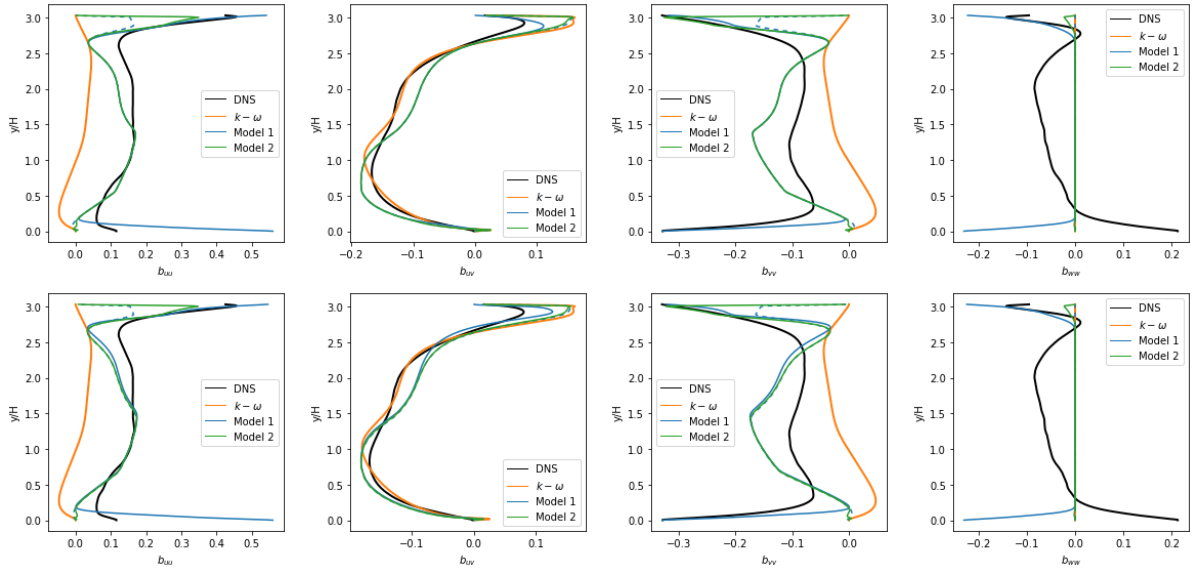
$$\mathcal{M}_1 = -3.191 T_2. \tag{6.9}$$

For the library method, the best model for the same library but multiplied with the damping functions obtained by the constrained STLSQ problem is

$$\mathcal{M}_2 = (-7.84 + 1.5f + 3.14f^2)T_2 + 1.61\left(1 - f^{2.0}\right)T_3 \tag{6.10}$$

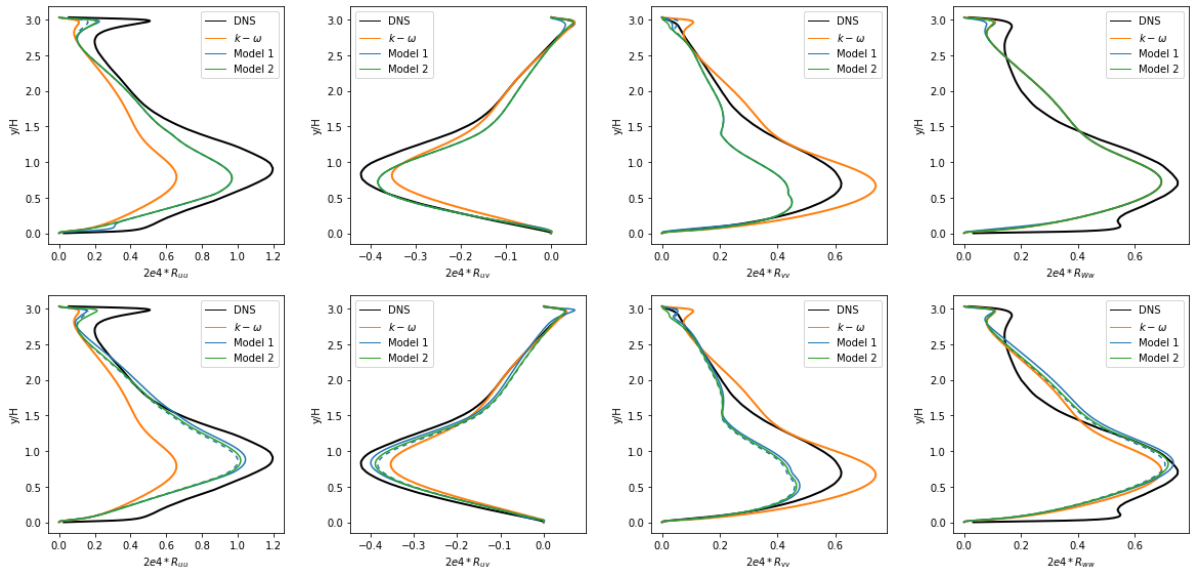
The predicted uu-, uv-, vv- and ww-components of the anisotropic stress tensor as obtained during training can be seen in the top row of Figure 6.16. The stresses are plotted for one cross section, $x = 4$, of the channel. In the inner part of the channel, both models predict the stresses similarly. At the top wall, $y = 3$, model 1 is better capable to predict the near-wall behaviour according the DNS data than its undamped base model, despite the small over-predictions of b_{uu} and b_{ww}. At the lower wall, $y = 0$, separation of the flow occurs and the approximations for b_{uu} and b_{ww} of model 1 are incorrect. However, the prediction of b_{vv} of model 1 is superior compared to its base model and model 2. Model 2 is not able to follow the DNS data of any component at the lower wall. At the top wall it improves the predictions of the diagonal components compared to the $k - \omega$ model, but not for b_{uv}. The bottom row of Figure 6.16 shows the propagation results of the anisotropic stress components. The predictions are very similar as before the propagation.

In the top row of Figure 6.17, the predicted components of the Reynolds stress tensor before propagation are shown. The predictions of both models overlap for all the components. Hence, both models improve the predictions of the uu- and uv- Reynolds stress components compared to the $k - \omega$ model and perform similar for the ww-component. Both models have more trouble in predicting R_{vv} in the inner part of the channel but improve the prediction at the top wall region compared to $k - \omega$. The propagation results of the models can be viewed in the bottom row of Figure 6.17. For both models, the results of the predictions of the R_{uu}, R_{uv} and R_{ww} are improved during the propagation compared



(a) b_{uu} **(b)** b_{uv} **(c)** b_{vv} **(d)** b_{ww}

Figure 6.16: Profiles of the uu-, uv-, vv- and ww-components of the anisotropic Reynolds stress in the PH case at $x = 4$ as predicted by models 1 and 2 defined by equations 6.9 and 6.10, which include damping functions, compared to DNS data and the $k - \omega$ RANS model. Model 1 consists of a base model (dashed line) to which damping is added manually. The top and bottom rows represent the stresses obtained before and after propagating the models in OpenFOAM, respectively. In the top row, the stresses predicted by both models are overlapping in the inner part of the channel.

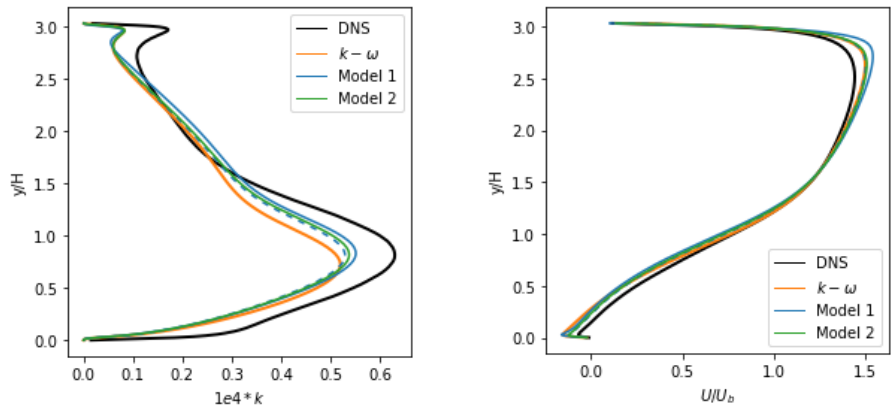


(a) R_{uu} **(b)** R_{uv} **(c)** R_{vv} **(d)** R_{ww}

Figure 6.17: Profiles of the uu-, uv-, vv- and ww-Reynolds stress components in the PH case at $x = 4$ as predicted by models 1 and 2 defined by equations 6.9 and 6.10, which include damping functions, compared to DNS data and the $k - \omega$ RANS model. Model 1 consists of a base model (dashed line) to which damping is added manually. The top and bottom rows represent the stresses obtained before and after propagating the models in OpenFOAM, respectively. In the top row, the stresses predicted by both models are overlapping.

to before. Compared to the undamped base of model 1, both damped models approximate the DNS data slightly better, which can indicate a minor effect of the wall damping functions. In general, the improvements of the near-wall behaviour of the Reynolds stress is limited for both models in the two-dimensional case containing separation of the flow. The very small difference between the propagation results of model 1 and its undamped base model suggest that this case suffers less from scaling issues. Also the kinetic energy of model 1 has not been changed much compared to its undamped base model, illustrated in Figure 6.18a. Therefore, the stresses are not scaled. Additionally, rescaling the Reynolds stresses is not easy for separated flows, because of the dynamic value of u_τ through the channel.

The predicted velocity profiles of the models are not improved compared the $k - \omega$ model as can be seen in Figure 6.18b. Model 1 even shows an over-prediction of the streamwise velocity near the upper-wall of the channel. A similar behaviour is observed when tensor $T_{ij}^{(1)}$ is present in a model, such as the model defined in equation 6.4 and plotted in Figure 6.6a. Model 1 and the model containing $T_{ij}^{(1)}$ both have an improved prediction of anisotropic shear stress at the upper wall, visible in Figures 6.16b and 6.6d. However, the shear Reynolds stresses are under-predicted by both models before propagation but over-predicted during the propagation of the models as can be seen in Figures 6.17b and 6.6c. In the rest of the domain, the shear anisotropic stress and Reynolds stress are not under-estimated by model 1, but are under-estimated by the model containing $T_{ij}^{(1)}$. This can explain why the velocity is not over-predicted in the rest of the channel by the damped model, but is by models containing $T_{ij}^{(1)}$. Both results suggest that the value of R_{uv} is important for the prediction of the streamwise velocity.



(a) (b)

Figure 6.18: Profiles of the kinetic energy and streamwise velocity in the PH case at $x = 4$ as predicted by models 1 and 2 defined by equations 6.9 and 6.10, which include damping functions, compared to DNS data and the $k - \omega$ RANS model. Model 1 consists of a base model (dashed line) to which damping is added manually.

7

Conclusions and Recommendations

Sparse symbolic regression is used to discover models for the nonlinear anisotropic Reynolds stress. As the modeling ansatz a nonlinear generalisation of the linear eddy viscosity model is used, expressed by a linear combination of invariant basis tensors multiplied by undetermined coefficients. For the coefficients, functional expressions are learned by solving a sparse regression problem, which involve a library of prior defined candidate functions.

A full model discovery pipeline is developed for discovering and testing data-driven turbulence models applied on three different test cases. First, DNS data is collected from literature, as well as the data of a corresponding $k-\omega$ RANS simulation. Features are extracted from the RANS data and a library of candidate functions is created. A sparse regression problem is derived and solved to select the best fitting functions from the library. The discovered model is propagated into the CFD solver OpenFOAM to obtain the predicted velocity profiles.

Most of the discovered models are able to improve the predictions of the Reynolds stress and the anisotropic stress over the $k-\omega$ RANS simulation. However, only a small number of the discovered models are able to slightly improve the predictions of the streamwise velocity compared to the $k-\omega$ model. Moreover, a considerable amount of the discovered model predicts the streamwise velocity worse than $k-\omega$. A different observation is done in the research of Schmelzer and Dwight [15]. The models obtained by their sparse symbolic regression method showed a consistent improvements of the velocity predictions. However, they included a second corrective model in the $k-\omega$ transport equations. Therefore it is recommended to expand the modeling framework to discover models for the k and ω equations as well.

The convenience of the open-box methodology of sparse symbolic regression is experienced when analysing the discovered models. It is observed that a specific basis tensor, $T_{ij}^{(1)}$, is always incorporated in models which predict the velocity poorly. Tensor $T_{ij}^{(1)}$ has the largest influence on the Reynolds shear stress and the kinetic energy, which apparently has a negative effect on the velocity prediction. A possible solution to improve the predictions of the velocity is to exclude tensor $T_{ij}^{(1)}$ from the library of candidate functions. A deeper analysis of the physical properties of the discovered models has not been done often in literature that the author is aware of. In the research of Weatheritt and Sandberg [48], they analysed which variables are active in the discovered models but did not relate them to the performance of the models. It is recommended to better investigate the reasons for the unsatisfying performance of models containing $T_{ij}^{(1)}$.

A majority of the models, including the ones which predict the streamwise velocity the best, do not generate a converged solution. When the propagation does not converge, deficiencies are visible in the flow fields, which are referred to as instabilities. Unconverged simulations are a general problem in data-driven turbulence modeling and multiple researches mention this problem such as [15] and [48]. However, not much attention is given to solve or understand the problem. In this project it is observed

that in some case the stability of propagated models improves when the relaxation factor is lowered, which is a parameter used in the solving algorithm in OpenFOAM. Unfortunately, this did not work for all cases. Another possible solution to improve stability is to use transient solvers, which are often more stable than the steady state solvers used in this research. However, this is not further investigated in the project. Future research should focus more on the causes and solutions of instabilities of propagated models, since this can give more insight in the robustness of data-driven turbulence modeling.

To answer the research question if there is a favourable regression method, four different regression problems are used to discover models, which include the LASSO, Enet, STLSQ and SR3 regression problems. Each regressor selects different functions from the library and hence, discovers different models. The Enet and SR3 regressors discovered the models which predict the velocity the best. These regression methods have one additional regularisation parameters compared to the LASSO and STLSQ problems. The models discovered by the STLSQ problem perform the most stable. That is, the models discovered by the STLSQ problem have the least amount of poor predictions for the streamwise velocity. However, the model performances highly depend on the selection of tensor $T_{ij}^{(1)}$. This complicates a direct comparison of the performances of the different regression problems. Furthermore, the effect of normalising the library before solving the regression problem is investigated. The normalisation has a different effect on each regression method. In general, normalisation improves the predictions for the stresses, while non-normalisation of the library stimulates generalisability. In previous research, mostly the LASSO and Enet regression methods are used. We suggest that the SR3 and STLSQ regressors are good methods as well. The choice of regression method rather depends on the values of the candidate functions and if normalisation of the library will be employed.

To prevent the model for producing infeasible values of the anisotropic stress tensor, constraints are added to the STLSQ and SR3 regression problem. In this way more physical knowledge is included in the modeling framework. It is observed that the constraints effectively force realisability of the models. This does not immediately lead to better predictions of the models or more converged simulations. Forcing realisability on the models is also done in the research of Ling *et al.* [12]. However, in this research, forcing realisability is done after the model discovery by specific transformations and no comments are made on the effect of this on the model performances. In our research it is also observed that realisable models may become unrealisable when propagated in the CFD solver. Forcing realisability of the models during propagation in a CFD solver could be investigated in future research.

Besides including physical knowledge by applying constraints to regression problems, additional physical features are included in the library. This is related to the research question if sparse symbolic regression is able to select the most relevant features. It is observed that the models, which predict the streamwise velocity the best, contain some of the physical features. However, it is not quantitatively investigated if these features are indeed the most relevant. Furthermore, the use of a wall damping function is investigated to model the near-wall region more accurately. Damping functions are commonly used in turbulence model but they are never applied in data-driven turbulence modeling, as far the author is aware of. Two strategies for including a wall damping function are tested. The first strategy consists of adding the damping function manually after discovering a base model. In the second strategy, damping functions are included in the library of candidate functions. In case of a simple channel flow, both strategies are able to model the near-wall region better than the baseline RANS model and discovered models trained without damping. For a case which incorporates flow separation, both model strategies showed limitations in the predictions of the flow variables in the region where flow separation occurs. The results suggest that using damping function in a data-driven modeling approach is promising but further improvements can be made.

To answer the last research question about the performance of the discovered turbulence models in terms of robustness and generalisability, cross validation is carried out in order to identify the models with the best predictive capacities. Hence, every model identified on one case are tested on the other cases as well. However, the three test cases used in this research are very similar to each other, since each case features separation of a flow in a channel. Therefore, the assessment of the generalisability of the models is limited. In general, we observe that models perform similarly on the three cases. That is, models performing good on one case, perform similarly on the other two cases. Generalisability

is partly improved when constraints are added to the regression problem and when the library is not normalised before solving the regression problem. For a better assessment of the generalisability of the models, future research should consider test cases with various flow features.

References

[1] F. T. M. Nieuwstadt, B. J. Boersma, and J. Westerweel, *Turbulence Introduction to Theory and Applications of Turbulent Flows*. 2016. DOI: 10.1007/978-3-319-31599-7.

[2] S. Rodriguez and P. Tools, *Applied Computational Fluid Dynamics and Turbulence Modeling*. 2019. DOI: 10.1007/978-3-030-28691-0.

[3] A. J. Smits, *Lectures in Fluid Mechanics Viscous Flows and Turbulence*, 2009.

[4] F. G. Schmitt, "About Boussinesq's turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity," *Comptes Rendus - Mecanique*, vol. 335, no. 9-10, pp. 617–627, Sep. 2007, ISSN: 16310721. DOI: 10.1016/j.crme.2007.08.004.

[5] S. Wallin and A. V. Johansson, "An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows," *Journal of Fluid Mechanics*, vol. 403, pp. 89–132, Jan. 2000. DOI: 10.1017/S0022112099007004.

[6] B. Eisfeld, C. Rumsey, and V. Togiti, "Verification and validation of a second-moment-closure model," in *AIAA Journal*, vol. 54, American Institute of Aeronautics and Astronautics Inc., 2016, pp. 1524–1541. DOI: 10.2514/1.J054718.

[7] T. B. Gatski and C. G. Speziale, "On Explicit Algebraic Stress Models for Complex Turbulent Flows," *Journal of Fluid Mechanics*, vol. 254, pp. 59–78, 1993. DOI: 10.1017/S0022112093002034.

[8] S. Luo, J. Cui, M. Vellakal, *et al.*, "Review and Examination of Input Feature Preparation Methods and Machine Learning Models for Turbulence Modeling," *arXiv*, 2020. DOI: 10.48550/ARXIV.2001.05485.

[9] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, pp. 477–508, 2020. DOI: 10.1146/annurev-fluid-010719-060214.

[10] K. Duraisamy, "Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence," *Physical Review Fluids*, vol. 6, May 2021. DOI: 10.1103/PhysRevFluids.6.050504.

[11] A. Beck and M. Kurz, "A perspective on machine learning methods in turbulence modeling," *GAMM Mitteilungen*, vol. 44, Mar. 2021. DOI: 10.1002/gamm.202100002.

[12] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, Nov. 2016. DOI: 10.1017/jfm.2016.615.

[13] J. Wu, H. Xiao, R. Sun, and Q. Wang, "RANS Equations with Explicit Data-Driven Reynolds Stress Closure Can Be Ill-Conditioned," *Journal of Fluid Mechanics*, vol. 869, pp. 553–586, Mar. 2019. DOI: 10.1017/jfm.2019.205.

[14] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 15, pp. 3932–3937, 2016. DOI: 10.1073/pnas.1517384113.

[15] M. Schmelzer, R. P. Dwight, and P. Cinnella, "Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression," *Flow, Turbulence and Combustion*, vol. 104, no. 2-3, pp. 579–603, Mar. 2020. DOI: 10.1007/s10494-019-00089-x.

[16] S. Beetham and J. Capecelatro, "Formulating turbulence closures using sparse regression with embedded form invariance," *Physical Review Fluids*, vol. 5, Aug. 2020. DOI: 10.1103/PhysRevFluids.5.084611.

[17] S. A. Orszag, "Analytical theories of turbulence," *Journal of Fluid Mechanics*, vol. 41, pp. 363–386, Apr. 1970. DOI: 10.1017/S0022112070000642.

[18] K. Duraisamy, G. Iaccarino, and H. Xiao, “Annual Review of Fluid Mechanics: Turbulence Modeling in the Age of Data,” *Annu. Rev. Fluid Mech*, vol. 51, pp. 357–377, 2019. DOI: 10.1146/annurev-fluid-010518.

[19] M. Reggente, “Statistical Gas Distribution Modelling for Mobile Robot Applications,” Ph.D. dissertation, Orebro University, 2014. DOI: 10.13140/2.1.1260.5760.

[20] D. C. Wilcox, *Turbulence modeling for CFD*. DCS Industries, 1993.

[21] S. B. Pope, *Turbulent Flows*. Cambridge University Press, Aug. 2000, ISBN: 9780521591256. DOI: 10.1017/CBO9780511840531.

[22] L. Prandtl, “Uber ein neues Formelsystem fur die ausgebildete Turbulenz,” *Nachrichten der Akademie der Wissenschaften zu G{\"o}ttingen, Mathematisch-Physikalische Klasse*, pp. 6–19, 1945.

[23] M. L. Kaandorp and R. P. Dwight, “Data-driven modelling of the Reynolds stress tensor using random forests with invariance,” *Computers and Fluids*, vol. 202, Apr. 2020. DOI: 10.1016/j.compfluid.2020.104497.

[24] J. L. Lumley, “Computational Modeling of Turbulent Flows,” *Advances in Applied Mechanics*, vol. 18, pp. 123–176, Jan. 1979. DOI: 10.1016/S0065-2156(08)70266-7.

[25] J. L. Lumley and G. R. Newman, “The return to isotropy of homogeneous turbulence,” *Journal of Fluid Mechanics*, vol. 82, pp. 161–178, Aug. 1977. DOI: 10.1017/S0022112077000585.

[26] M. Emory and D. G. Iaccarino, *Visualizing turbulence anisotropy in the spatial domain with componentality contours*, 2014.

[27] W. P. Jones and B. E. Launder, “The prediction of laminarization with a two-equation model of turbulence,” *International Journal of Heat and Mass Transfer*, vol. 15, pp. 301–314, Feb. 1972. DOI: 10.1016/0017-9310(72)90076-2.

[28] W. Rodi and N. N. Mansour, “Low Reynolds number k—ϵ modelling with the aid of direct simulation data,” *Journal of Fluid Mechanics*, vol. 250, pp. 509–529, 1993.

[29] E. R. v. Driest, “On Turbulent Flow Near a Wall,” *Journal of the Aeronautical Sciences*, vol. 23, no. 11, pp. 1007–1011, 1956. DOI: 10.2514/8.3713.

[30] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of Research and Development*, vol. 3, pp. 206–226, 1959. DOI: 10.1147/rd.33.0210.

[31] Z. Zhou, *Machine Learning*. Singapore: Springer Nature Singapore, 2021, ISBN: 978-981-15-1966-6. DOI: 10.1007/978-981-15-1967-3.

[32] T. Kohonen, “An Introduction to Neural Computing,” *Neural Networks*, vol. 1, pp. 3–16, 1988. DOI: 10.1016/0893-6080(88)90020-2.

[33] T. K. Ho, “Random decision forests,” in *Proceedings of the International Conference on Document Analysis and Recognition*, vol. 1, IEEE Computer Society, 1995, pp. 278–282. DOI: 10.1109/ICDAR.1995.598994.

[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning; Data Mining, Inference, and Prediction*, Second. Springer Series in Statistics, 2009. DOI: 10.1007/978-0-387-84858-7.

[35] C. Ferreira, “Gene Expression Programming: a New Adaptive Algorithm for Solving Problems,” *Complex System*, vol. 13, 2001. DOI: 10.48550/arxiv.cs/0102027.

[36] J. Weatheritt and R. Sandberg, “A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship,” *Journal of Computational Physics*, vol. 325, pp. 22–37, 2016. DOI: 10.1016/j.jcp.2016.08.015.

[37] S. L. Brunton, “Applying Machine Learning to Study Fluid Mechanics,” *Acta Mechanica Sinica*, vol. 37, pp. 1718–1726, 2021. DOI: 10.1007/s10409-021-01143-6.

[38] W. N. Edeling, P. Cinnella, and R. P. Dwight, “Predictive RANS simulations via Bayesian Model-Scenario Averaging,” *Journal of Computational Physics*, vol. 275, pp. 65–91, 2014. DOI: 10.1016/j.jcp.2014.06.052.

[39] J. Ray, S. Lefantzi, S. Arunajatesan, and L. Dechant, "Bayesian Parameter Estimation of a k-ϵ Model for Accurate Jet-in-Crossflow Simulations," *AIAA Journal*, vol. 54, no. 8, pp. 2432–2448, 2016. DOI: 10.2514/1.J054758.

[40] B. fabritius, "Application of Genetic Algorithms to Problems in Computational Fluid Dynamics," Ph.D. dissertation, University of Exeter, Jan. 2014.

[41] B. Tracey, K. Duraisamy, and J. J. Alonso, "A machine learning strategy to assist turbulence model development," in *53rd AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics Inc, AIAA, 2015. DOI: 10.2514/6.2015-1287.

[42] M. Gamahara and Y. Hattori, "Searching for turbulence models by artificial neural network," *Physical Review Fluids*, vol. 2, Jun. 2017. DOI: 10.1103/PhysRevFluids.2.054604.

[43] R. Maulik and O. San, "A neural network approach for the blind deconvolution of turbulent flows," *Journal of Fluid Mechanics*, vol. 831, pp. 151–181, 2017. DOI: 10.1017/jfm.2017.637.

[44] A. Vollant, G. Balarac, and C. Corre, "Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures," *Journal of Turbulence*, vol. 18, no. 9, pp. 854–878, Sep. 2017. DOI: 10.1080/14685248.2017.1334907.

[45] C. Xie, K. Li, C. Ma, and J. Wang, "Modeling subgrid-scale force and divergence of heat flux of compressible isotropic turbulence by artificial neural network," *Physical Review Fluids*, vol. 4, 2019. DOI: 10.1103/PhysRevFluids.4.104605.

[46] A. Beck, D. Flad, and C. D. Munz, "Deep neural networks for data-driven LES closure models," *Journal of Computational Physics*, vol. 398, 2019. DOI: 10.1016/j.jcp.2019.108910.

[47] J. Ling and J. Templeton, "Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty," *Physics of Fluids*, vol. 27, Aug. 2015. DOI: 10.1063/1.4927765.

[48] J. Weatheritt and R. D. Sandberg, "The development of algebraic stress models using a novel evolutionary algorithm," *International Journal of Heat and Fluid Flow*, vol. 68, pp. 298–318, 2017. DOI: 10.1016/j.ijheatfluidflow.2017.09.017.

[49] J.-X. Wang, J.-L. Wu, and H. Xiao, "A Physics Informed Machine Learning Approach for Reconstructing Reynolds Stress Modeling Discrepancies Based on DNS Data," *Physical Review Fluids*, vol. 2, no. 3, Mar. 2017. DOI: 10.1103/PhysRevFluids.2.034603.

[50] Y. Zhao, H. D. Akolekar, J. Weatheritt, V. Michelassi, and R. D. Sandberg, "RANS turbulence model development using CFD-driven machine learning," *Journal of Computational Physics*, vol. 411, Mar. 2020. DOI: 10.1016/j.jcp.2020.109413.

[51] M. Reissmann, J. Hasslberger, R. D. Sandberg, and M. Klein, "Application of Gene Expression Programming to a-posteriori LES modeling of a Taylor Green Vortex," *Journal of Computational Physics*, vol. 424, Sep. 2021. DOI: 10.1016/j.jcp.2020.109859.

[52] J. Zhong, L. Feng, and Y. S. Ong, "Gene Expression Programming: A Survey," *IEEE Computational Intelligence Magazine*, vol. 12, pp. 54–72, Aug. 2017. DOI: 10.1109/MCI.2017.2708618.

[53] H. Hmida, S. B. Hamida, A. Borgi, and M. Rukoz, "Scale Genetic Programming for large Data sets: Case of Higgs Bosons Classification," in *Procedia Computer Science*, vol. 126, Elsevier B.V., Sep. 2018, pp. 302–311. DOI: 10.1016/j.procS.2018.07.264.

[54] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Nov. 2019. DOI: 10.1016/j.jcp.2018.10.045.

[55] H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations," *Physics of Fluids*, vol. 34, no. 7, Jul. 2021. DOI: 10.1063/5.0095270.

[56] S. Taghizadeh, F. Witherden, Y. Hassan, and S. Girimaji, "Turbulence closure modeling with data-driven techniques: Investigation of generalizable deep neural networks," *Physics of Fluids*, vol. 33, no. 11, Nov. 2021. DOI: 10.1063/5.0070890.

[57] S. B. Pope, “A more general effective-viscosity hypothesis,” *Journal of Fluid Mechanics*, vol. 72, no. 2, p. 331, Mar. 1975. DOI: 10.1017/S0022112075003382.

[58] J. L. Wu, H. Xiao, and E. Paterson, “Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework,” *Physical Review Fluids*, vol. 3, Jul. 2018. DOI: 10.1103/PhysRevFluids.3.074602.

[59] R. Tibshirani, “Regression Shrinkage and Selection Via the Lasso,” *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267–288, Jan. 1996. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[61] A. Kaptanoglu, B. de Silva, U. Fasel, *et al.*, “PySINDy: A comprehensive Python package for robust sparse system identification,” *Journal of Open Source Software*, vol. 7, p. 3994, 2022. DOI: 10.21105/joss.03994.

[62] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, “PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data,” *Journal of Open Source Software*, vol. 5, p. 2104, May 2020. DOI: 10.21105/joss.02104.

[63] P. Zheng, T. Askham, S. L. Brunton, *et al.*, “A Unified Framework for Sparse Relaxed Regularized Regression: SR3,” *IEEE*, vol. 7, pp. 1404–1423, 2019. DOI: 10.1109/ACCESS.2018.2886528.

[64] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz, “A unified sparse optimization framework to learn parsimonious physics-informed models from data,” *IEEE Access*, vol. 8, pp. 169 259–169 271, Sep. 2020. DOI: 10.1109/ACCESS.2020.3023625.

[65] M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe, “Interior-point methods for large-scale cone programming,” Tech. Rep.

[66] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP solver for embedded systems,” in *European Control Conference*, IEEE Computer Society, Jul. 2013, pp. 3071–3076. DOI: 10.23919/ecc.2013.6669541.

[67] S. Diamond and S. Boyd, “CVXPY: A Python-Embedded Modeling Language for Convex Optimization,” *Journal of Machine Learning Research*, vol. 17, pp. 1–5, Apr. 2016. DOI: 10.48550/ARXIV.1603.00943.

[68] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, “A tensorial approach to computational continuum mechanics using object-oriented techniques,” *Computers in Physics*, vol. 12, no. 6, pp. 620–631, Nov. 1998. DOI: 10.1063/1.168744.

[69] L. S. Caretto, A. D. Gosman, S. V. Patankar, and D. B. Spalding, “Two calculation procedures for steady, three-dimensional flows with recirculation,” in *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, H. Cabannes and R. Temam, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1973, pp. 60–68, ISBN: 978-3-540-38392-5.

[70] J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics*, 4th ed. Springer, 2020. DOI: 10.1007/978-3-319-99693-6.

[71] R. McConkey, E. Yee, and F. S. Lien, “A curated dataset for data-driven turbulence modelling,” *Scientific Data*, vol. 8, 2021. DOI: 10.1038/s41597-021-01034-2.

[72] H. Xiao, J. L. Wu, S. Laizet, and L. Duan, “Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations,” *Computers & Fluids*, vol. 200, 2020. DOI: 10.1016/J.COMPFLUID.2020.104431.

[73] J.-P. Laval and M. Marquillie, “Direct Numerical Simulations of Converging–Diverging Channel Flow,” in *Progress in Wall Turbulence: Understanding and Modeling*, M. Stanislas, J. Jimenez, and I. Marusic, Eds., vol. 14, Dordrecht: Springer Netherlands, 2011, pp. 203–209. DOI: 10.1007/978-90-481-9603-621.

[74] Y. Bentaleb, S. Lardeau, and M. A. Leschziner, “Large-eddy simulation of turbulent boundary layer separation from a rounded step,” *Journal of Turbulence*, vol. 13, 2012. DOI: 10.1080/14685248.2011.637923.

[75] R. D. Moser, J. Kim, and N. N. Mansour, “Direct numerical simulation of turbulent channel flow up to Re_τ=590,” *Physics of Fluids*, vol. 11, pp. 943–945, Mar. 1999. DOI: 10.1063/1.869966.

[76] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, “Pathwise coordinate optimization,” *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007. DOI: 10.1214/07-aoas131.

A

Coordinate Descent Algorithm

Written more generally the loss function of the LASSO problem is defined as

$$L(\beta) = ||y - X\beta||_2^2 + \lambda||\beta||_1, \tag{A.1}$$

which consists of the residual the sum of squares, $RSS = ||y - X\beta||_2^2$ and the penalty term $\lambda||\beta||_1$. This can also be written as

$$L(\beta) = \sum_{i=1}^{n}(y_i - \sum_{j=0}^{p} X_{ij}\beta_j)^2 + \lambda\sum_{j=0}^{p}|\beta_j| \tag{A.2}$$

The minimum of the loss function along one coordinate β_j is found by setting the partial derivatives to zero [76]. The partial derivative of RSS with respect to one coordinate β_j is

$$\begin{aligned}
\frac{\partial RSS}{\partial \beta_j} &= -2\sum_{i=1}^{n} X_{ij}(y_i - \sum_{j=0}^{p} X_{ij}\beta_j) \\
&= -2\sum_{i=1}^{n} X_{ij}(y_i - \sum_{k\neq j} X_{ik}\beta_k - X_{ij}\beta_j) \\
&= -2\sum_{i=1}^{n} X_{ij}(y_i - \sum_{k\neq j} X_{ik}\beta_k) + 2\beta_j\sum_{i=1}^{n} X_{ij}^2 \\
&= -2\rho_j + 2\beta_j z_j
\end{aligned} \tag{A.3}$$

Here, $\rho_j = -2\sum_{i=1}^{n} X_{ij}(y_i - \sum_{k\neq j} X_{ik}\beta_k)$ and $z_j = \sum_{i=1}^{n} X_{ij}^2$. The L_1 penalty is not continuously differentiable. Hence, subgradients are used to define

$$\lambda\frac{\partial|\beta_j|}{\partial\beta_j} = \begin{cases} -\lambda & \text{when} \quad \beta_j < 0 \\ [-\lambda, \lambda] & \text{when} \quad \beta_j = 0 \\ \lambda & \text{when} \quad \beta_j > 0 \end{cases} \tag{A.4}$$

Computing the subgradients of the LASSO loss function and equating to zero to find the minimum gives

$$\begin{aligned}
\frac{\partial L(\beta)}{\partial\beta_j} &= -2\rho_j + 2\beta_j z_j + \lambda\frac{\partial|\beta_j|}{\partial\beta_j} \\
0 &= \begin{cases} -2\rho_j + 2\beta_j z_j - \lambda & \text{when} \quad \beta_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda] & \text{when} \quad \beta_j = 0 \\ -2\rho_j + 2\beta_j z_j + \lambda & \text{when} \quad \beta_j > 0 \end{cases}
\end{aligned} \tag{A.5}$$

As solution for β_j we find

$$\begin{cases} \beta_j = \frac{\rho_j + \frac{\lambda}{2}}{z_j} & \text{for} \quad \rho_j < -\frac{\lambda}{2}, \\ \beta_j = 0 & \text{for} \quad \frac{\lambda}{2} \leq \rho_j \leq \frac{\lambda}{2}, \\ \beta_j = \frac{\rho_j - \frac{\lambda}{2}}{z_j} & \text{for} \quad \rho_j > -\frac{\lambda}{2}. \end{cases} \tag{A.6}$$

Equation A.6 is recognised as the soft thresholding function $\frac{1}{z_j} S(\rho_j, \lambda)$. Algorithm 1 represents the complete coordinate descent algorithm. The coordinate descent algorithm has the potential to be quite efficient. One full iteration can be completed at a computational cost of $\mathcal{O}(2np)$, since only two inner products are required in each iteration. However, the algorithm requires an unknown number of iterations.

Algorithm 1 Coordinate Descent Algorithm for LASSO

Initialise $\beta_j = 0$
repeat
  for $j = 0, 1, ..., p$ **do**
    $\rho_j \leftarrow -2 \sum_{i=1}^{n} X_{ij}(y_i - \sum_{k \neq j} X_{ik}\beta_k)$
    $z_j \leftarrow \sum_{i=1}^{n} X_{ij}^2$
    $\beta_j \leftarrow \frac{1}{z_j} S(\rho_j, \lambda)$
  end for
until convergence

B

Cholesky Decomposition

The Ridge regression estimator,

$$\arg\min_{\beta} ||y - X\beta||_2^2 + \lambda||\beta||_2^2, \tag{B.1}$$

is the solution to

$$(X^T X + \lambda I)\beta = X^T y. \tag{B.2}$$

The problem is solved by treating it as a linear system

$$Ax = b. \tag{B.3}$$

The Cholesky decomposition is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.

$$A = LL^* \tag{B.4}$$

An algorithm to compute the entries of matrix L can be derived by carrying out the matrix-matrix product LL^T and equating the results to the matrix A. This algorithm is listed in Algorithm 2, which computes the matrix L column-wise and stores the result in the lower triangular part of A.

Algorithm 2 Cholesky Factorisation Step

for $k = 1 \rightarrow n$ **do**
  $A(k,k) \leftarrow L(k,k) = \sqrt{A(k,k) - \sum_{j=1}^{k-1} C(k,j)^2}$
  for $i = k+1 \rightarrow n$ **do**
    $A(i,k) \leftarrow C(i,k) = \frac{1}{C(k,k)}(A(i,k) - \sum_{j=1}^{k-1} C(i,j)C(k,j))$
  end for
end for

The computational cost of the Cholesky factorisation is given by

$$\frac{1}{3}n^3 + \mathcal{O}(n^2)\text{flops}. \tag{B.5}$$

Once the system is factorised into $LL^*x = b$, the linear system can be solved by a forward and backward linear solve. The forward solves determines the auxiliary vector y of the system $Ly = b$. The backward determines the solution x of the system $L^*x = y$. The forward and backward substitution algorithm is stated in Algorithm 3.

The computational cost of the forward and backward substitution is given by

$$n^2 + \mathcal{O}(n)\text{flops}. \tag{B.6}$$

Algorithm 3 Forward-Backward Solving Algorithm

for $k = 1 \rightarrow n$ **do**
  $y(i) \leftarrow [b(i) - L(i, 1 : i - 1) \cdot y(1 : i - 1)]$
end for
for $i = n \rightarrow i$ **do**
  $x(i) \leftarrow [y(i) - L^*(i, i + 1 : n) \cdot u(i + 1 : n)]/L^*(i, j)$
end for

C

Discovered Models Physical Library

C.1. Best Models

Table C.1: Best performing models for the PH, CD and CBFS case discovered using the normalised physical library and the normalised mean squared errors of the predicted streamwise velocity for each case.

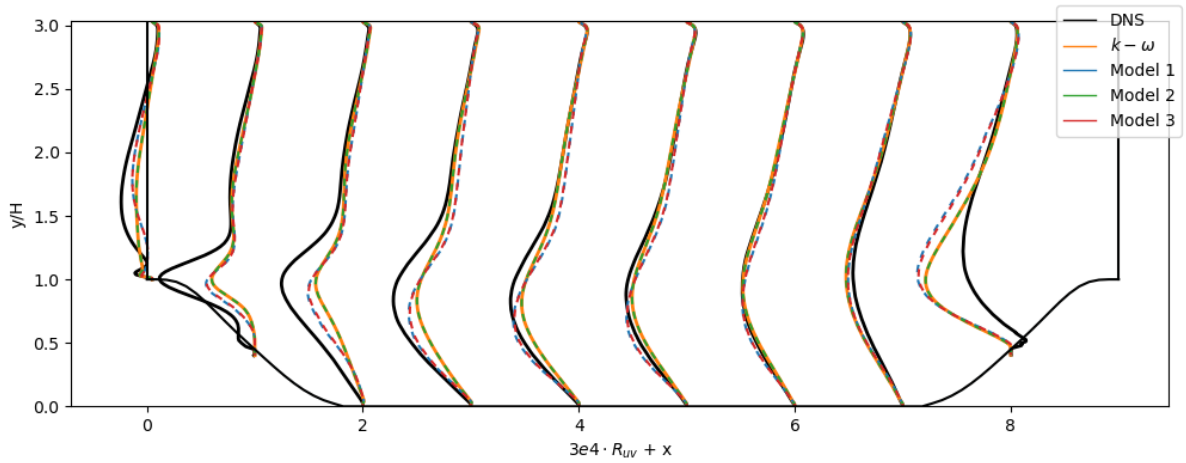
Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = (0.48I_2 - 4.7)T_2$	0.8801	1.1130	1.0000
$\mathcal{M}_2 = 15.1I_1T_3$	0.9714	0.9672	0.9742
$\mathcal{M}_3 = (1.2I_1 - 1.5I_2 + 3.96q_4 - 5.119)T_2$	1.0229	1.1534	0.9626

Table C.2: Best performing models for the PH, CD and CBFS case discovered using the not normalised physical library and the normalised mean squared errors of the predicted streamwise velocity for each case.

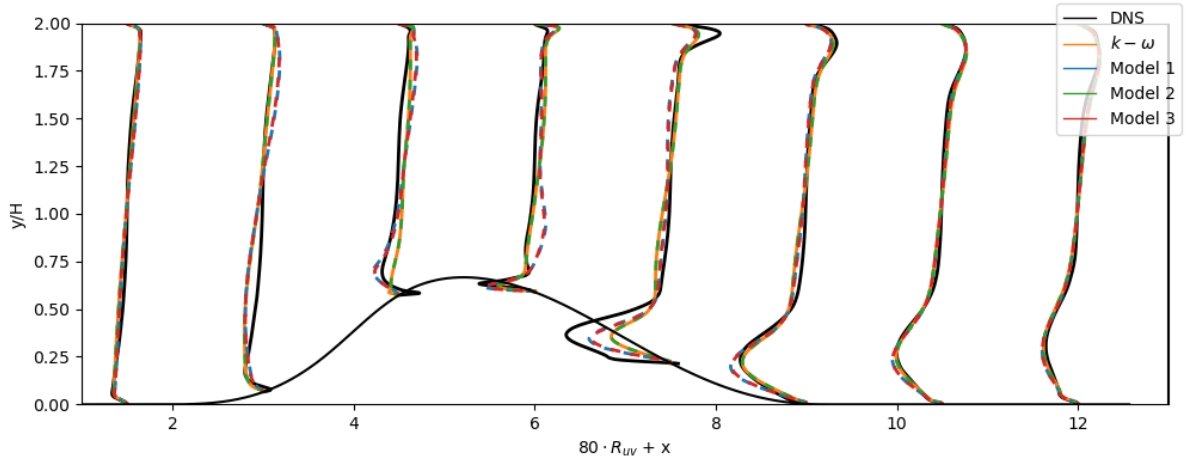
Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = 70.3I_2T_2 + (-1177.3I_1 - 1248.8I_2 - 226.5q_1)T_3$	0.5639	0.8002	0.9744
$\mathcal{M}_2 = 96.3I_2T_2 + (52.4I_1 - 136.6I_2 - 29.5q_4)T_3$	1.1470	0.7271	0.9137
$\mathcal{M}_3 = 96.3I_2T_2 + (-84.3I_1 - 154.4I_2)T_3$	1.0924	0.7274	0.9124

Table C.3: Best performing models for the PH, CD and CBFS case discovered using the constrained regression problems and not normalised physical library and the normalised mean squared errors of the predicted streamwise velocity for each case.

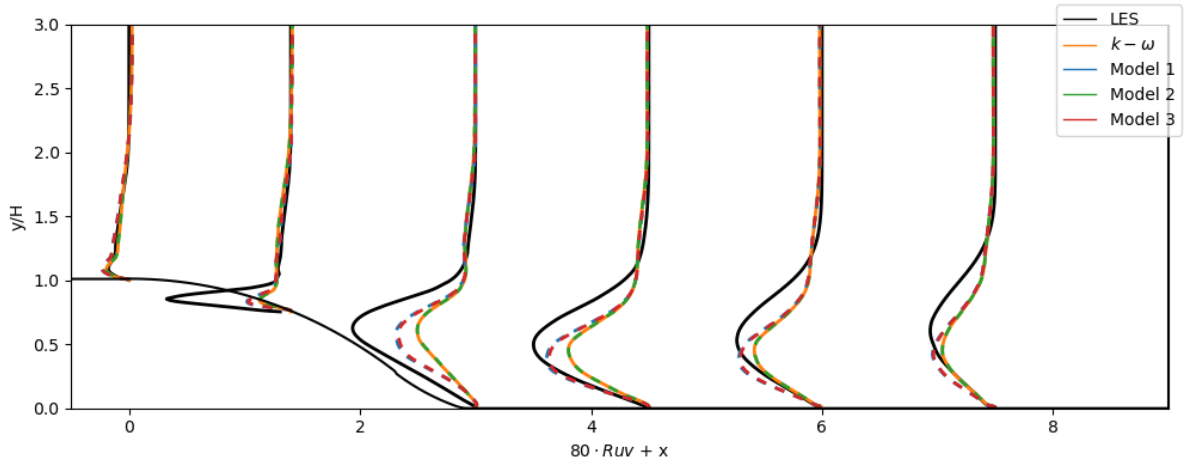
Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = 46.1I_2T_2 - 61.55q_1T_3$	0.7919	1.04557	1.1178
$\mathcal{M}_2 = 46.1I_2T_2 - 61.55q_1T_3$	0.7919	1.04557	1.1178
$\mathcal{M}_3 = (7.6q_4 + 5.4q_7 - 6.2)T_2$	0.9652	1.0707	0.9472



(a) PH

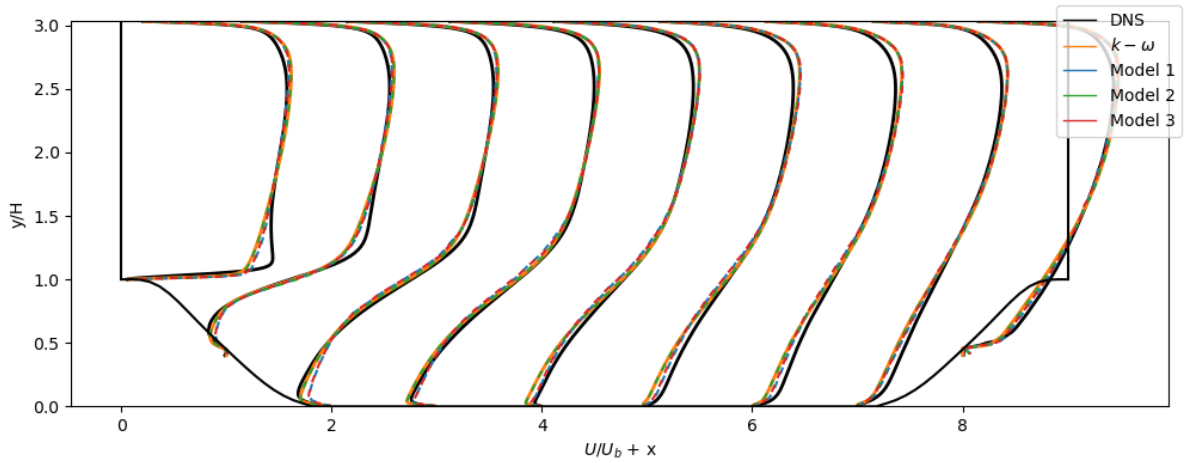


(b) CD

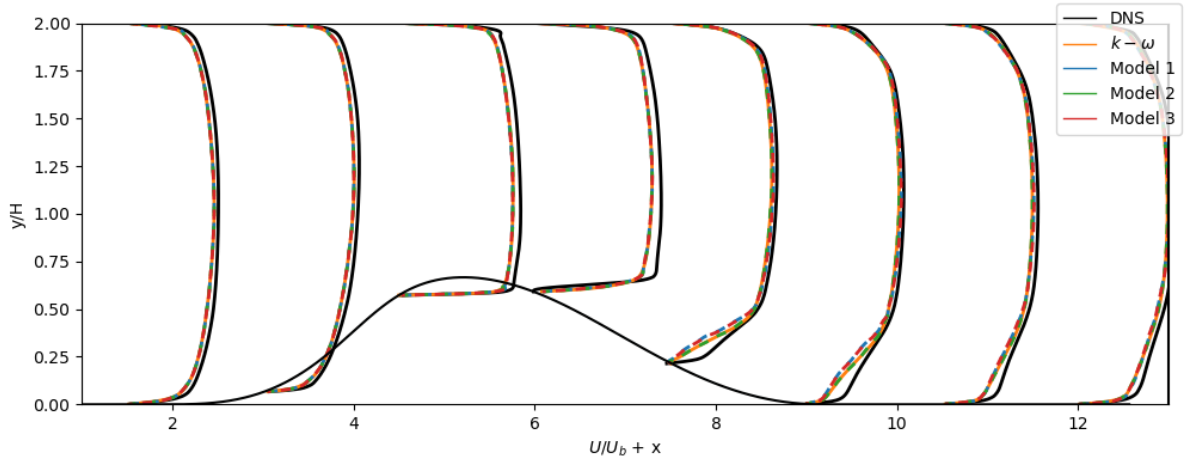


(c) CBFS

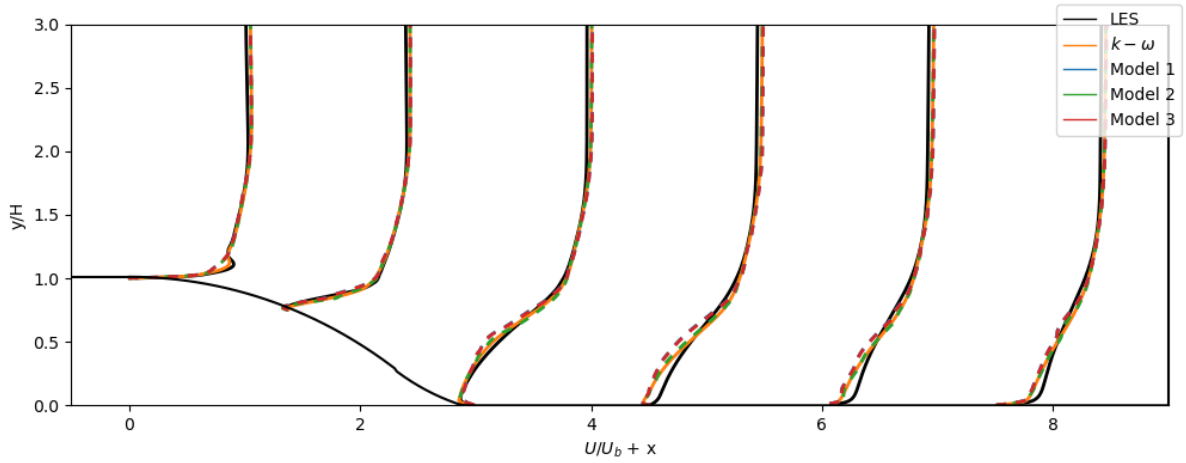
Figure C.1: Shear Reynolds stress predictions of the propagated best performing models (Table C.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

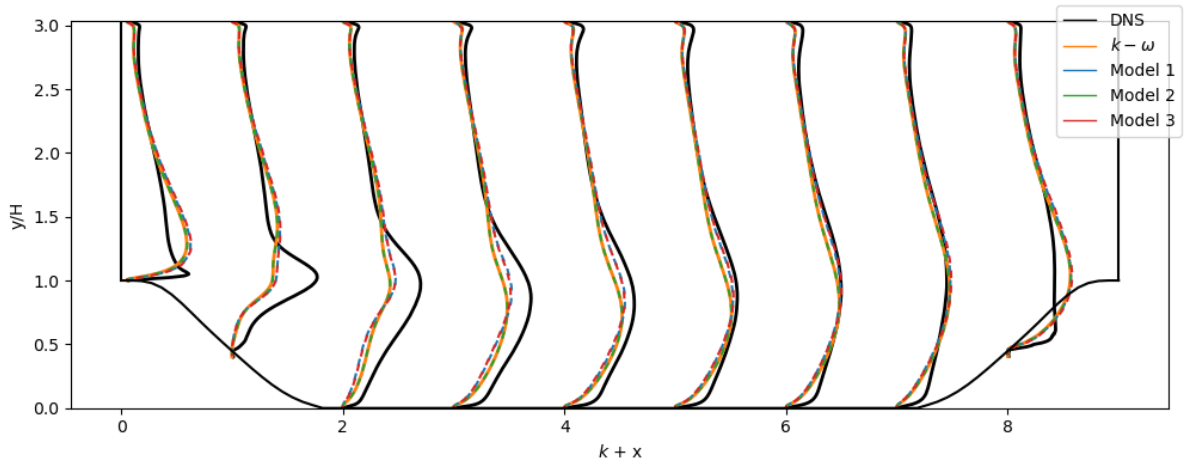


(b) CD

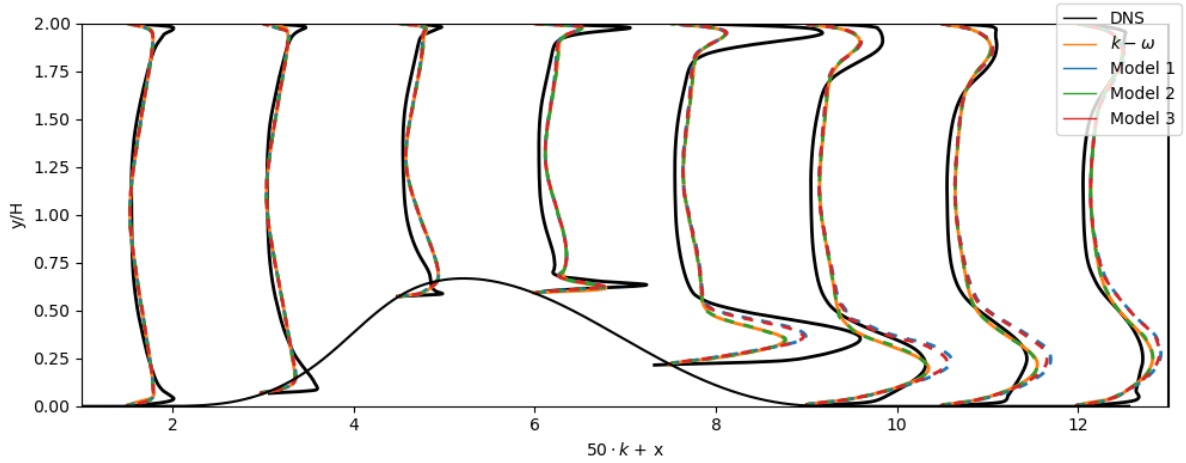


(c) CBFS

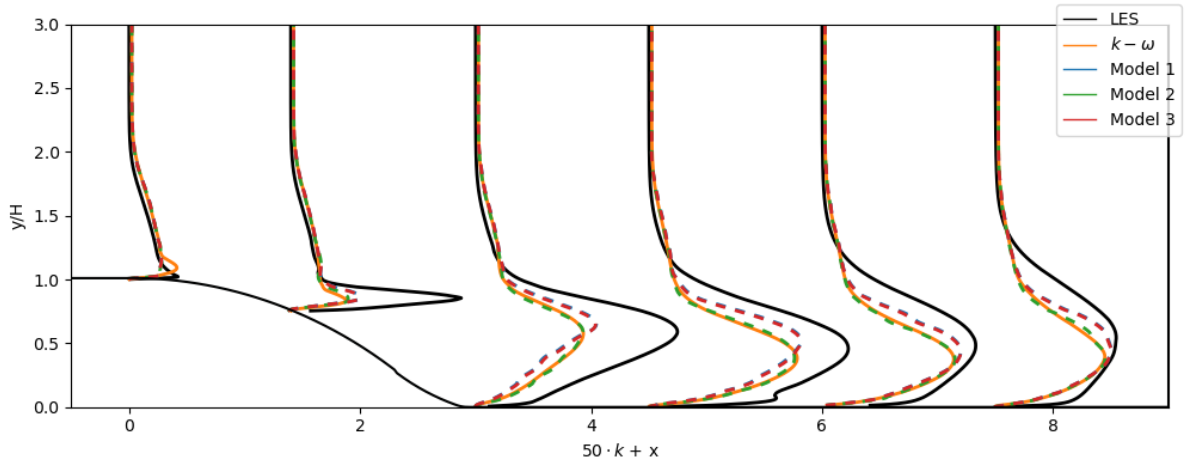
Figure C.2: Streamwise velocity predictions of the propagated best performing models (Table C.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

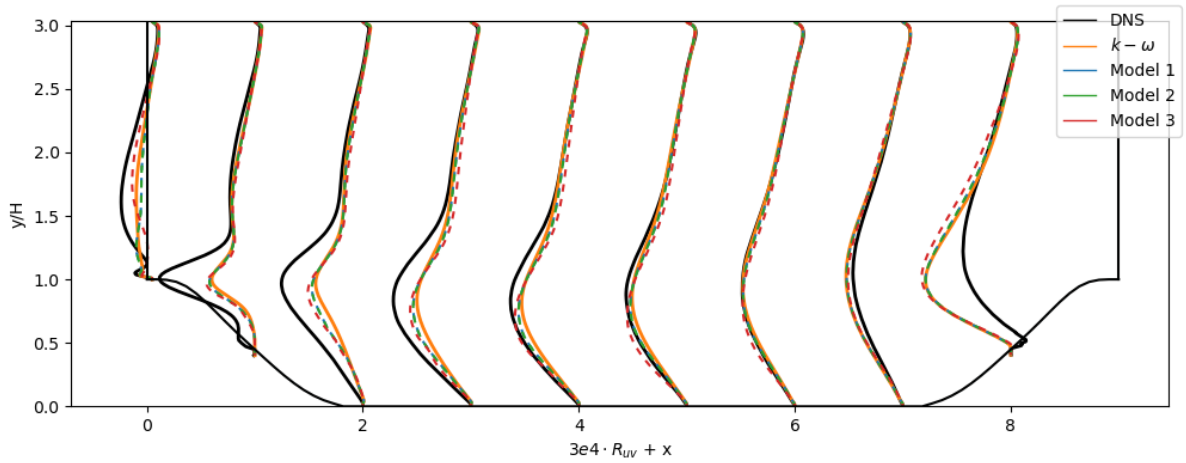


(b) CD

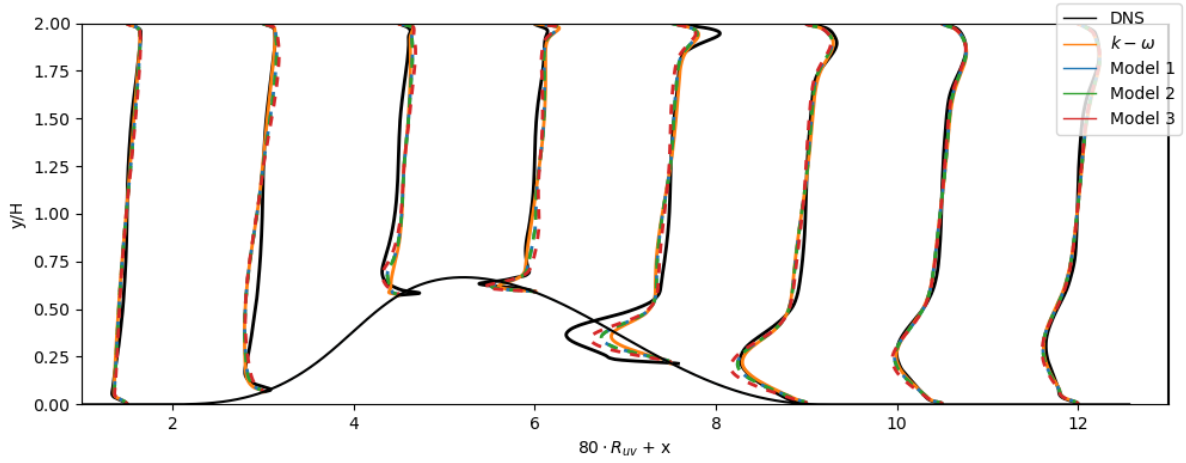


(c) CBFS

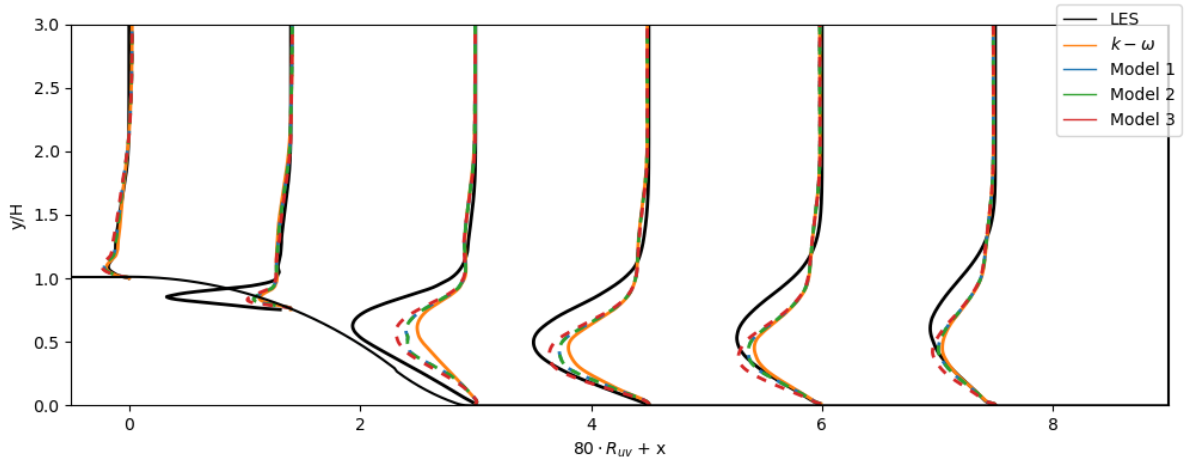
Figure C.3: Kinetic energy predictions of the propagated best performing models (Table C.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

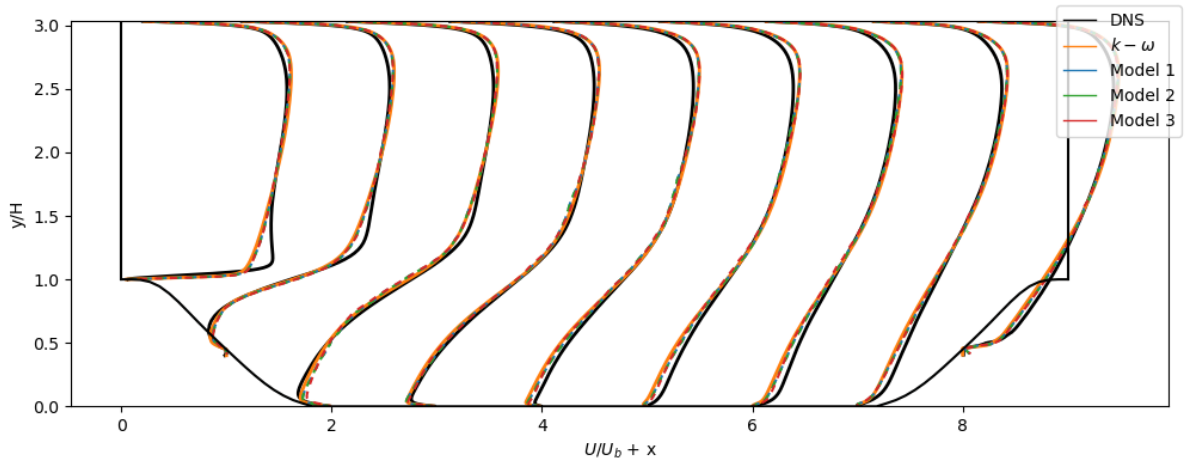


(b) CD

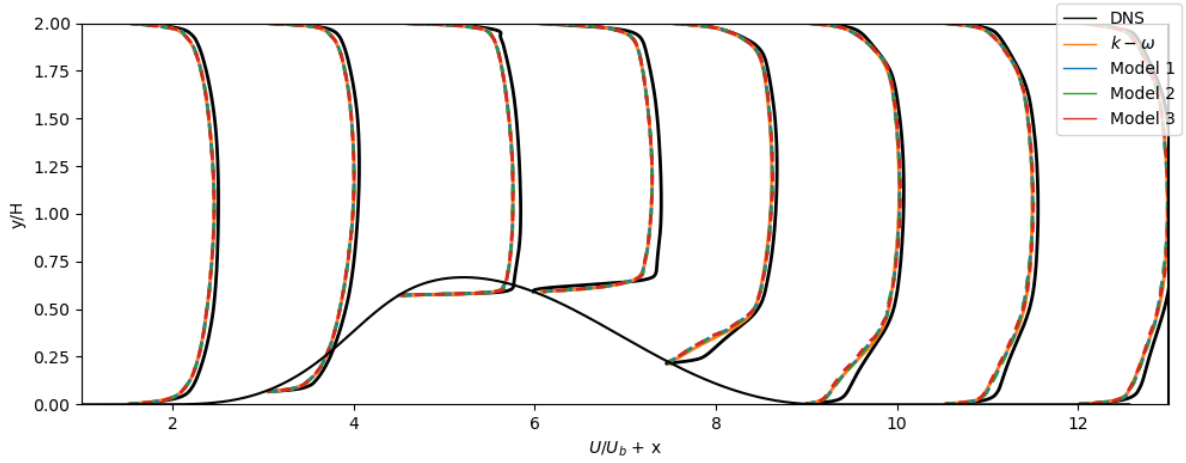


(c) CBFS

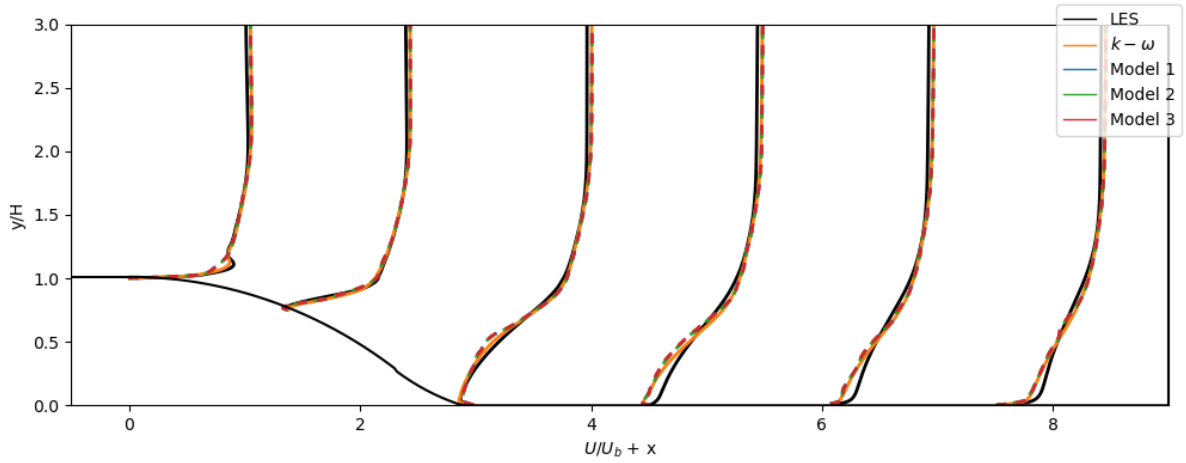
Figure C.4: Shear Reynolds stress predictions of the propagated best performing models (Table C.3) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

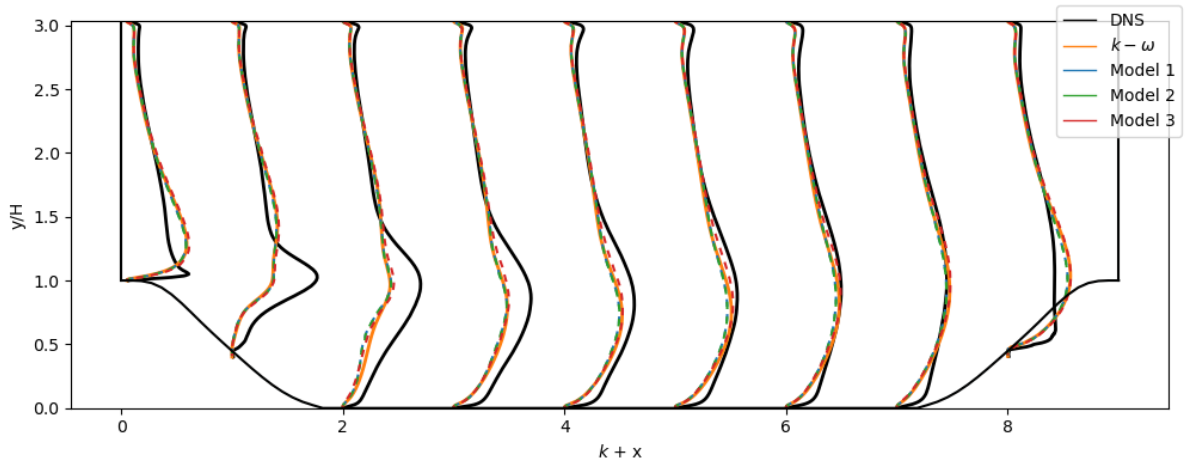


(b) CD

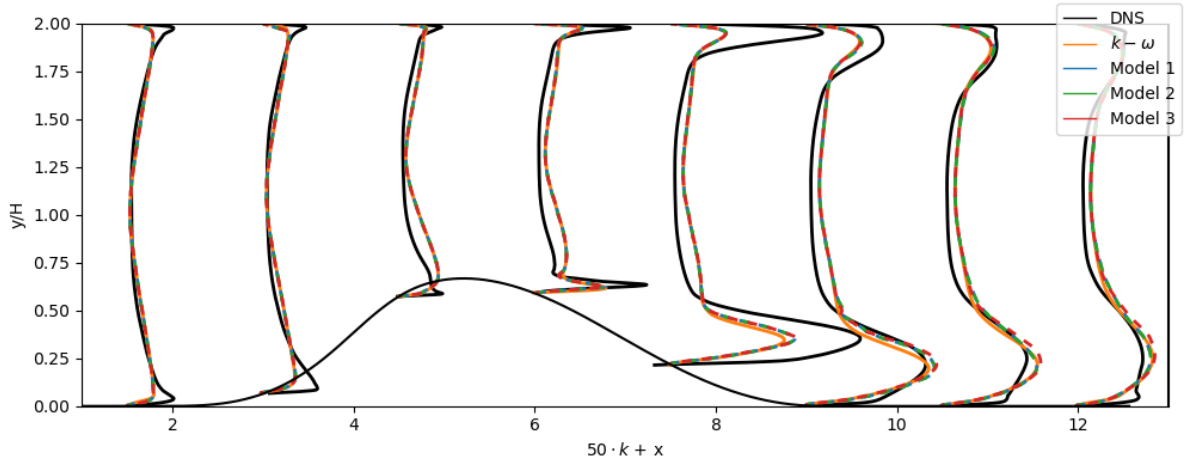


(c) CBFS

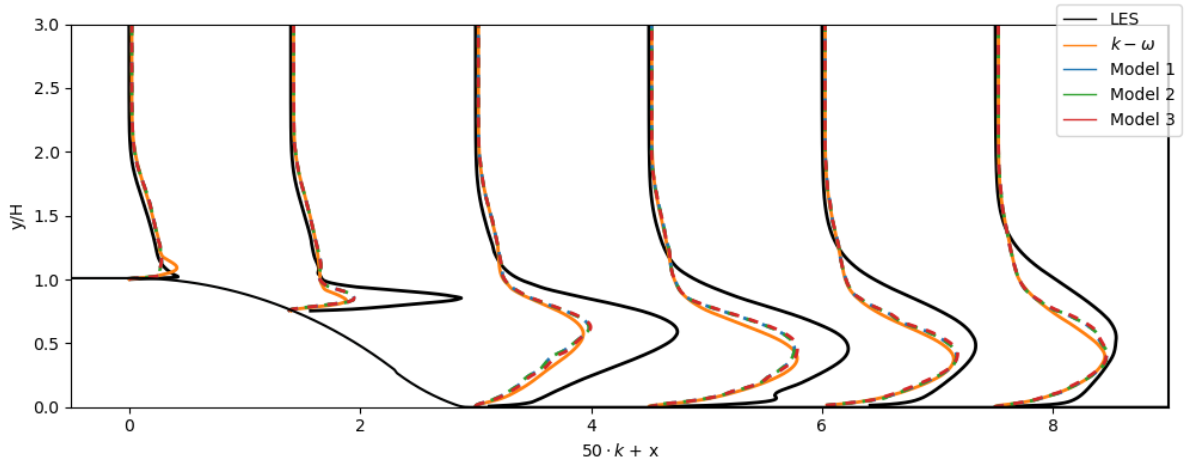
Figure C.5: Streamwise velocity predictions of the propagated best performing models (Table C.3) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH



(b) CD



(c) CBFS

Figure C.6: Kinetic energy predictions of the propagated best performing models (Table C.3) compared to DNS and $k-\omega$ RANS data for each case.

C.2. Discovered Models PH

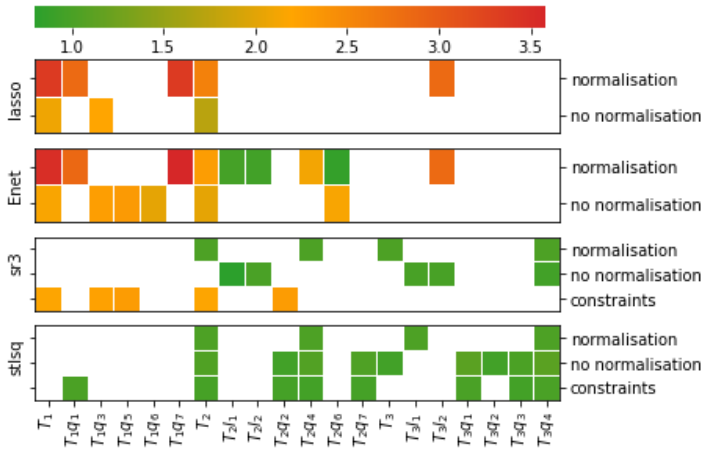
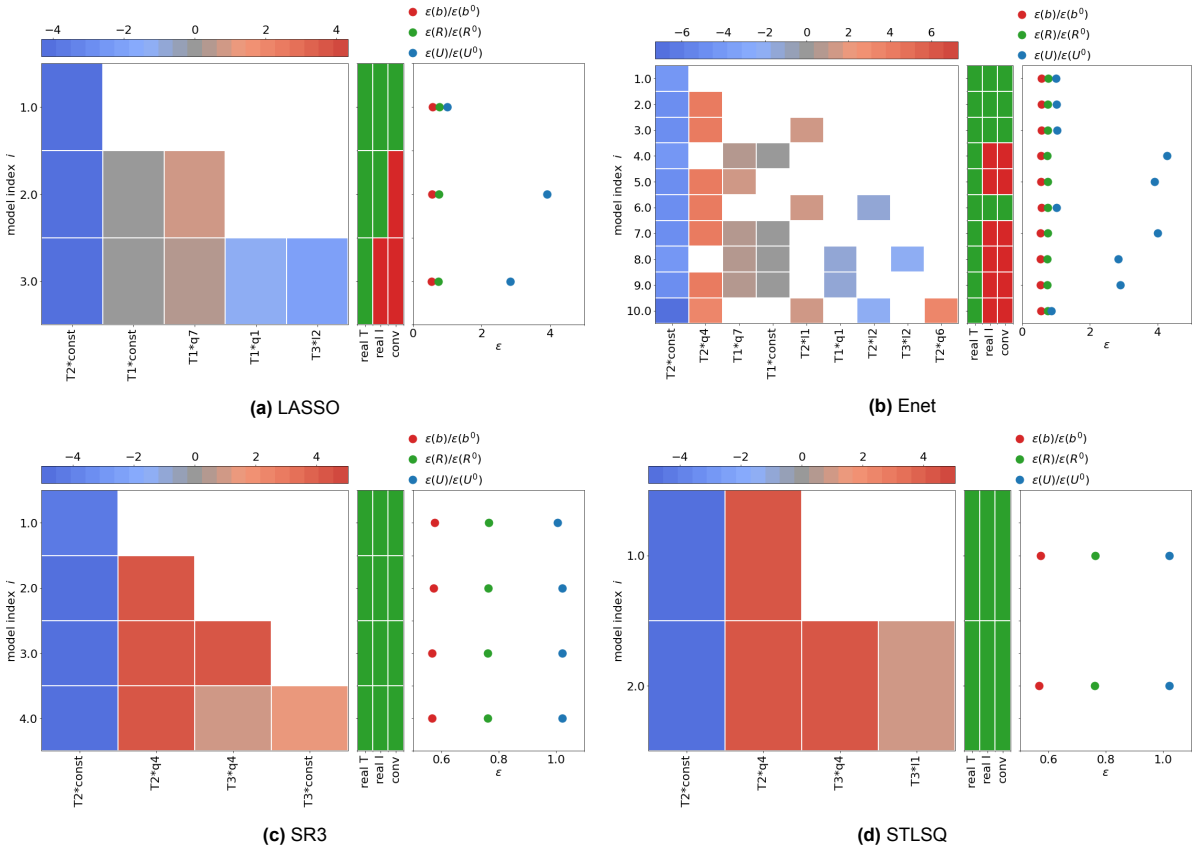


Figure C.7: Feature selection per regression problem and setting for the PH case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

Discovered Models with Normalisation



(a) LASSO

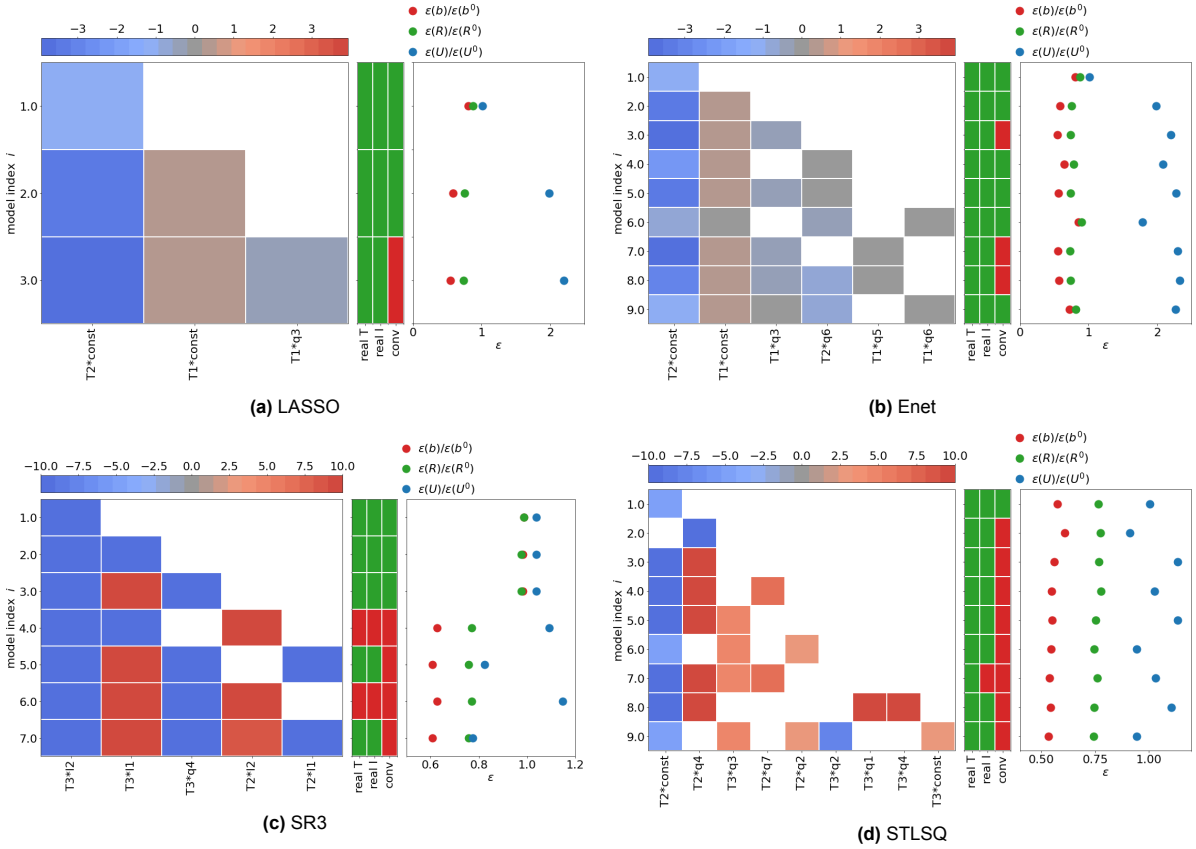
(b) Enet

(c) SR3

(d) STLSQ

Figure C.8: Discovered models by each regressor on the PH case using the normalised library.

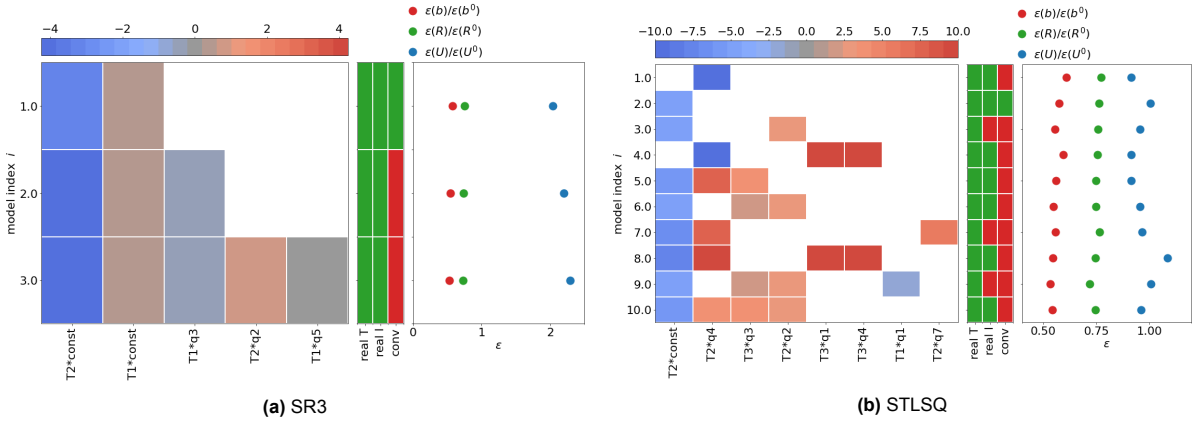
Discovered Models without Normalisation



(a) LASSO **(b)** Enet **(c)** SR3 **(d)** STLSQ

Figure C.9: Discovered models by each regressor on the PH case using the not normalised library.

Discovered Models with Constraints



(a) SR3 **(b)** STLSQ

Figure C.10: Discovered models by each constrained regressor on the PH case using the not normalised library.

C.3. Discovered Models CD

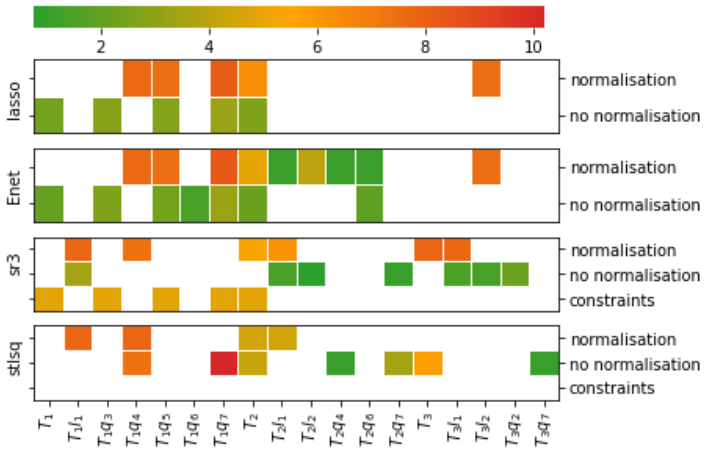
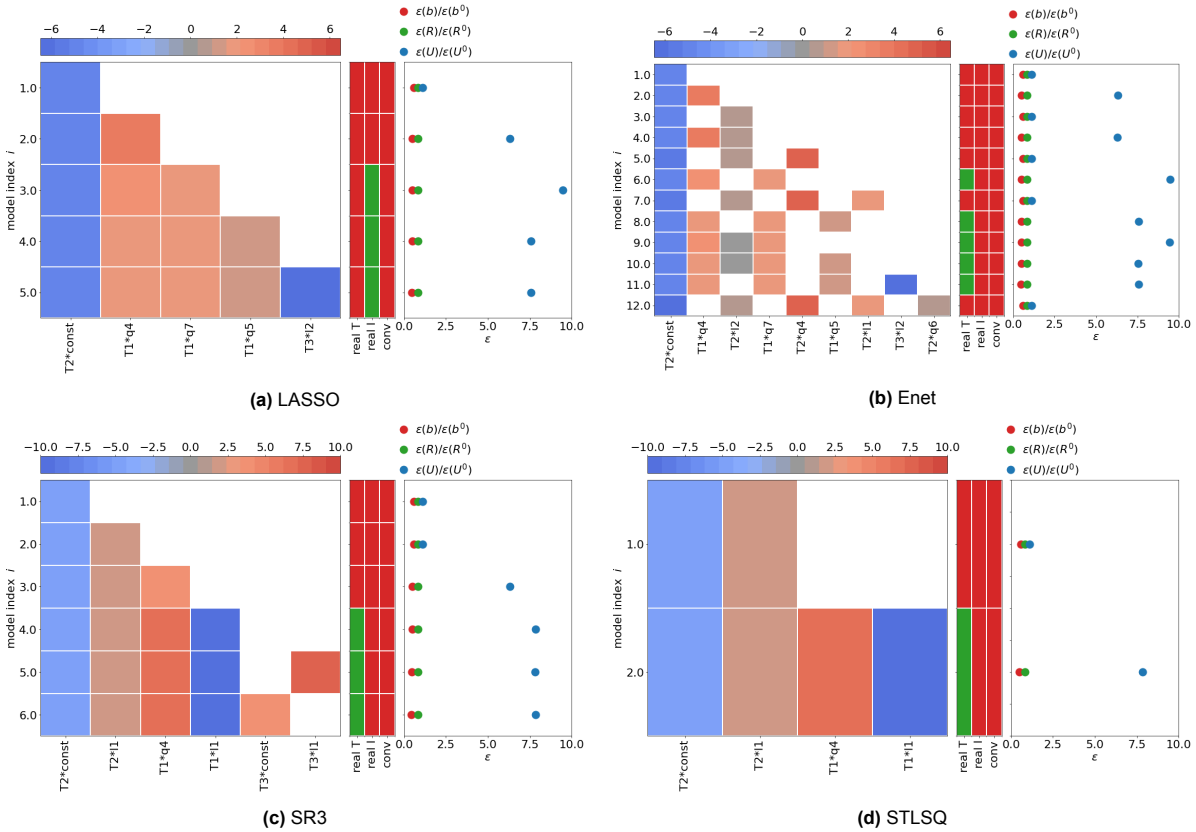


Figure C.11: Feature selection per regression problem and setting for the CD case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

Discovered Models with Normalisation



(a) LASSO

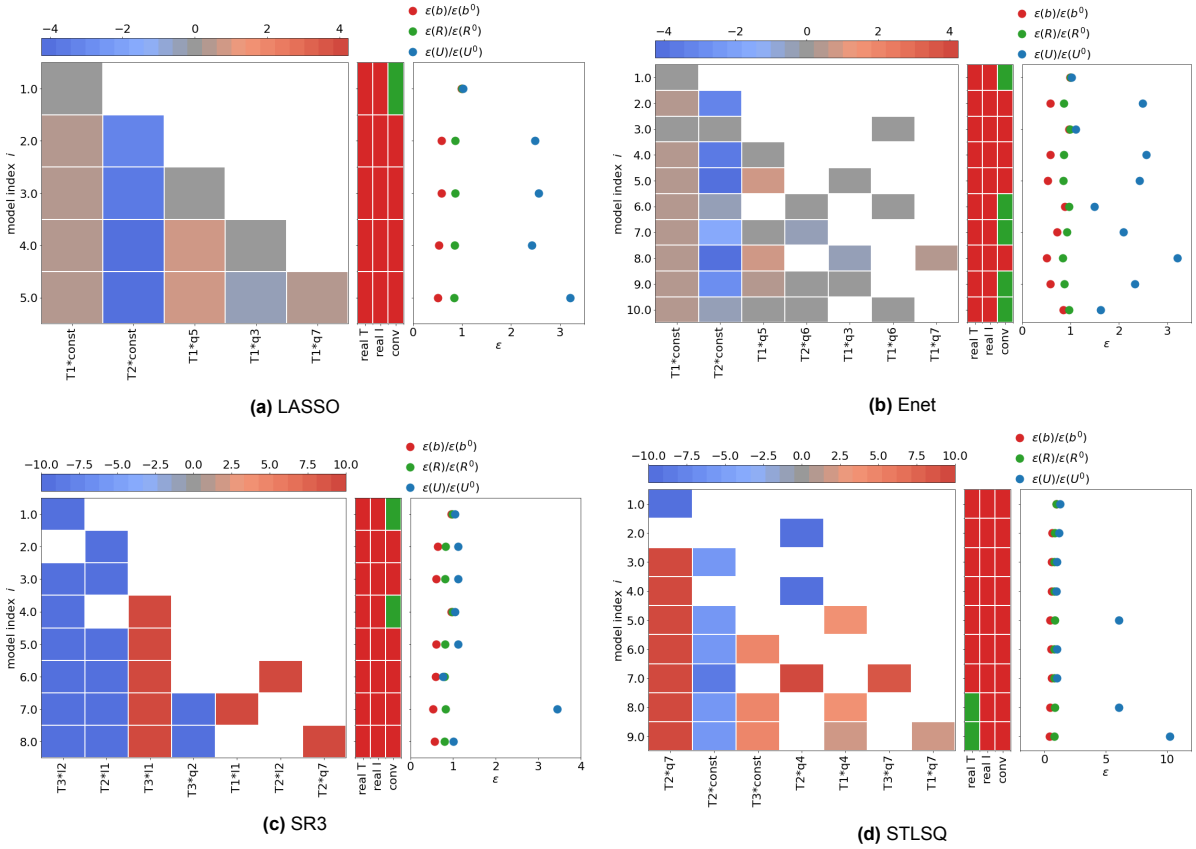
(b) Enet

(c) SR3

(d) STLSQ

Figure C.12: Discovered models by each regressor on the CD case using the normalised library.

Discovered Models without Normalisation

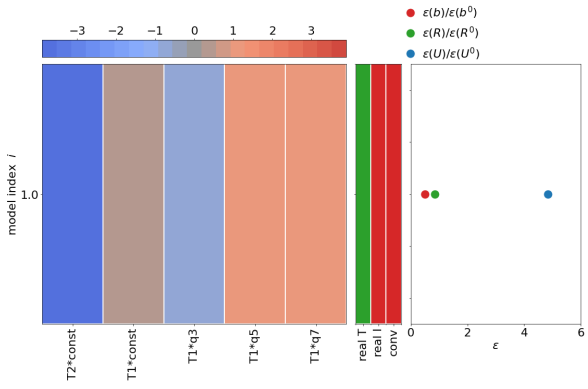


(a) LASSO **(b)** Enet

(c) SR3 **(d)** STLSQ

Figure C.13: Discovered models by each regressor on the CD case using the not normalised library.

Discovered Models with Constraints



(a) SR3

Figure C.14: Discovered models by each constrained regressor on the CD case using the normalised library.

C.4. Discovered Models CBFS

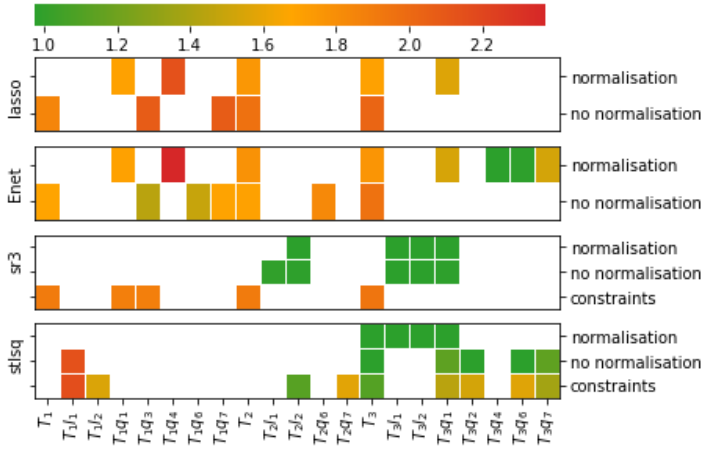
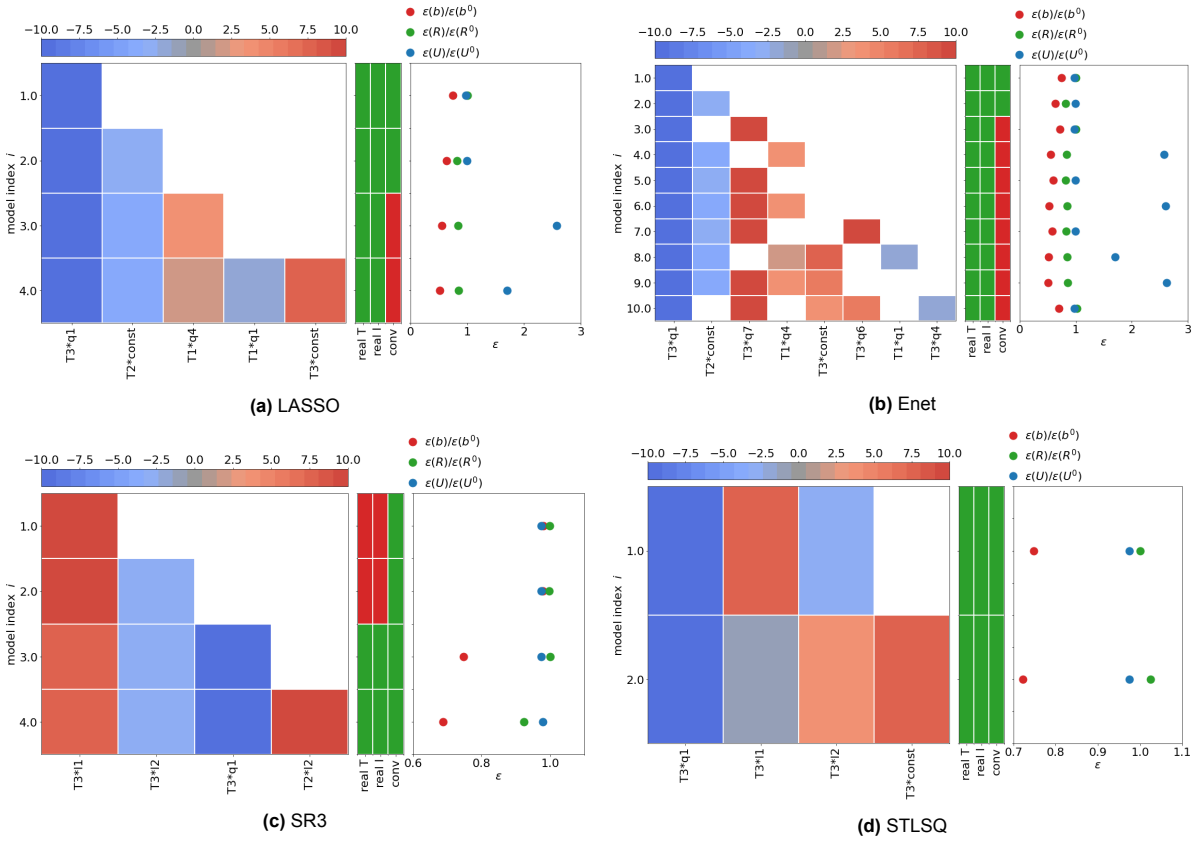


Figure C.15: Feature selection per regression problem and setting for the CBFS case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

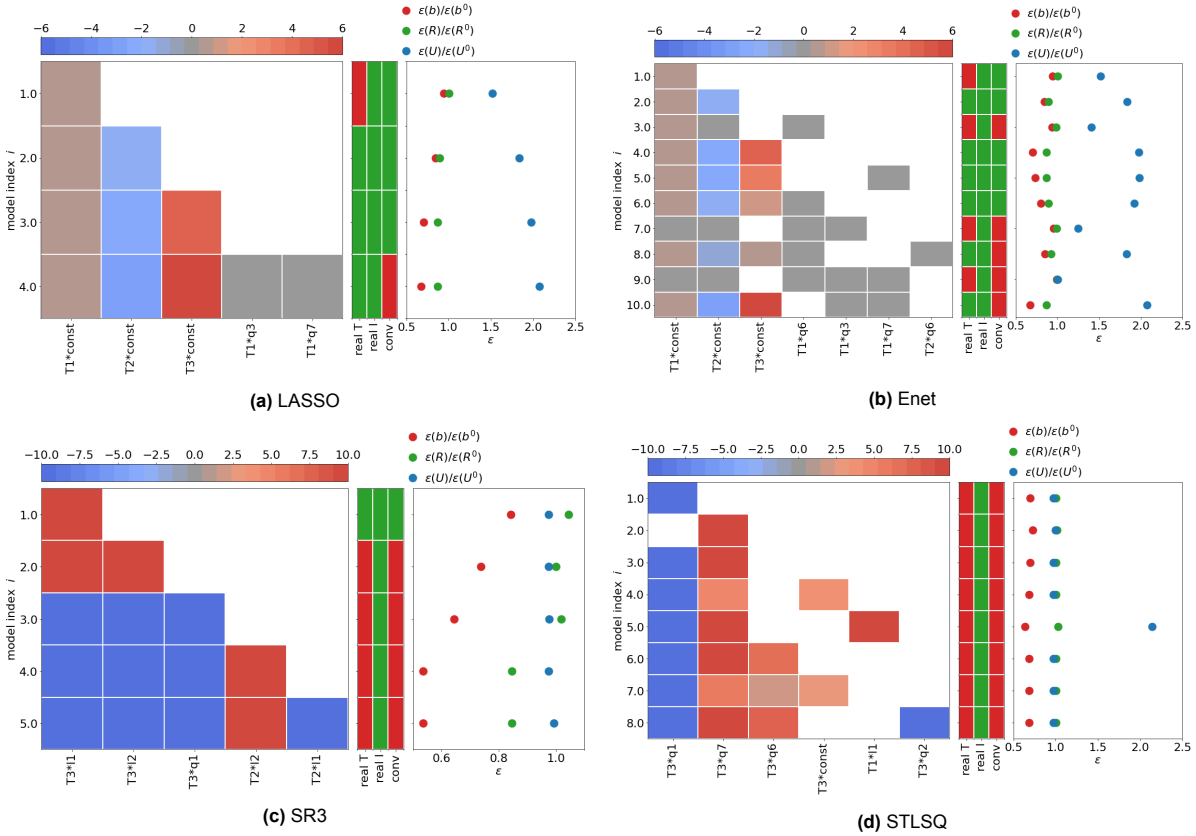
Discovered Models with Normalisation



(a) LASSO **(b)** Enet **(c)** SR3 **(d)** STLSQ

Figure C.16: Discovered models by each regressor on the CBFS case using the normalised library.

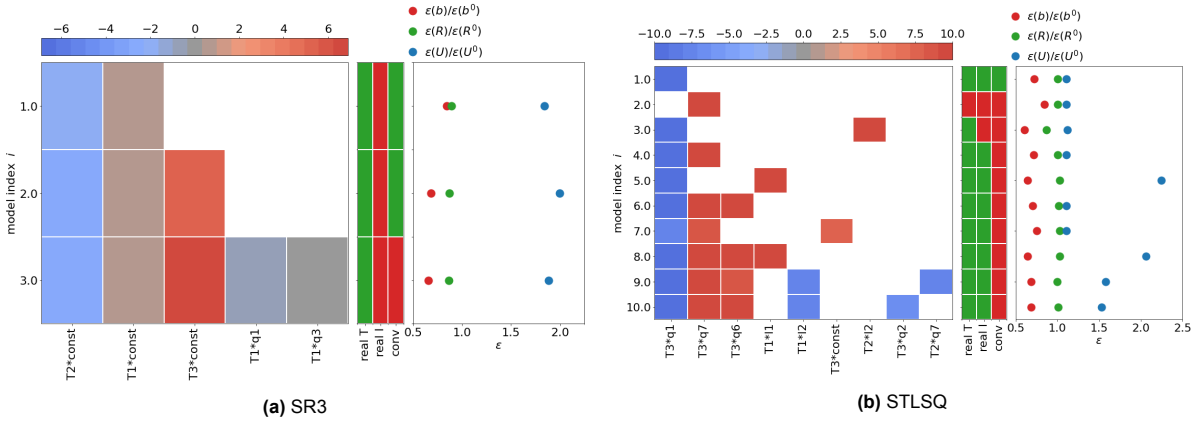
Discovered Models without Normalisation



(a) LASSO **(b)** Enet **(c)** SR3 **(d)** STLSQ

Figure C.17: Discovered models by each regressor on the CBFS case using the not normalised library.

Discovered Models with Constraints



(a) SR3 **(b)** STLSQ

Figure C.18: Discovered models by each constrained regressor on the CBFS case using the not normalised library.

C.5. Performances of Discovered Models

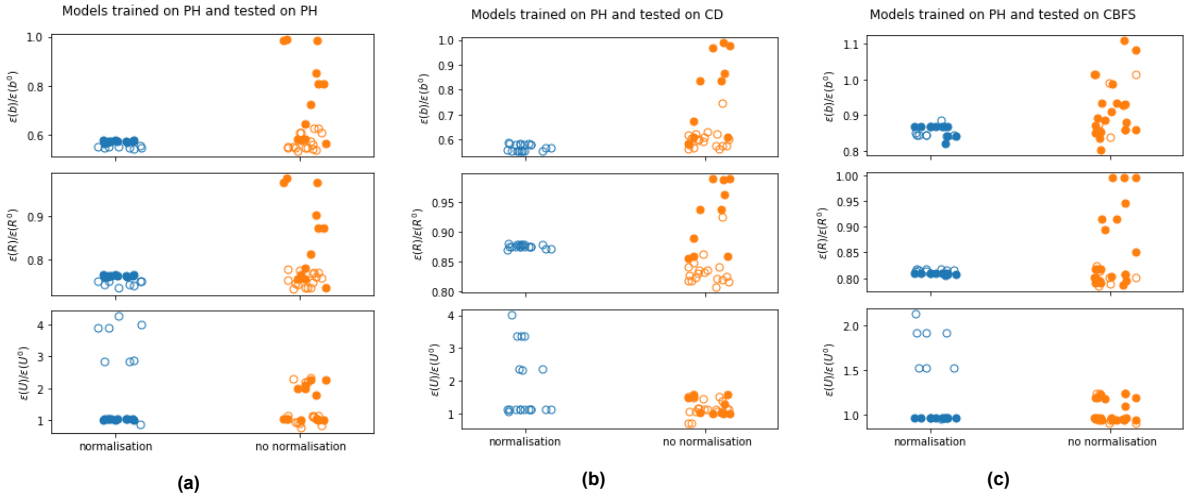


Figure C.19: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the PH case using a normalised or not normalised library and tested on the PH, CD and CBFS.

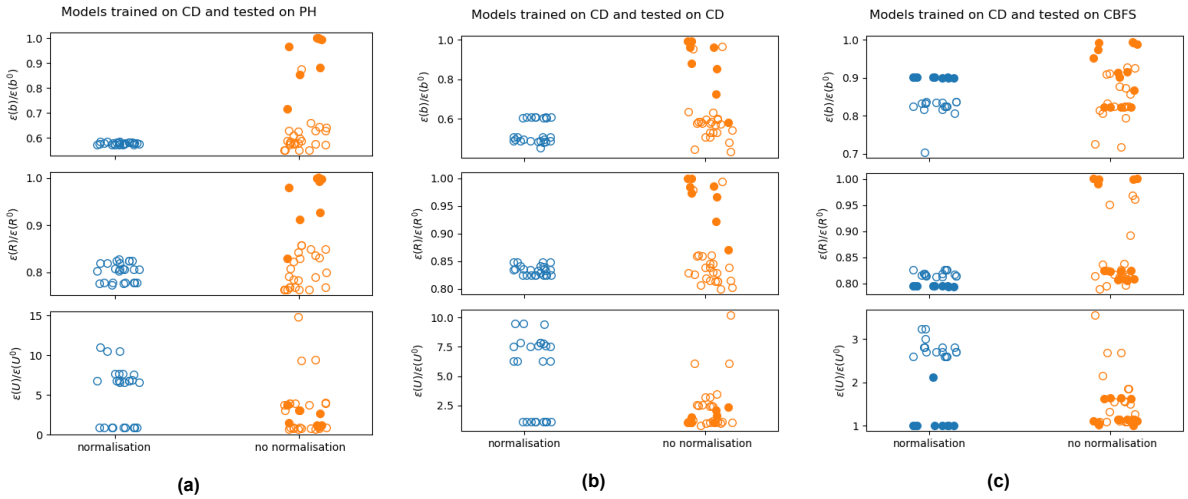


Figure C.20: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the CD case using a normalised or not normalised library and tested on the PH, CD and CBFS.

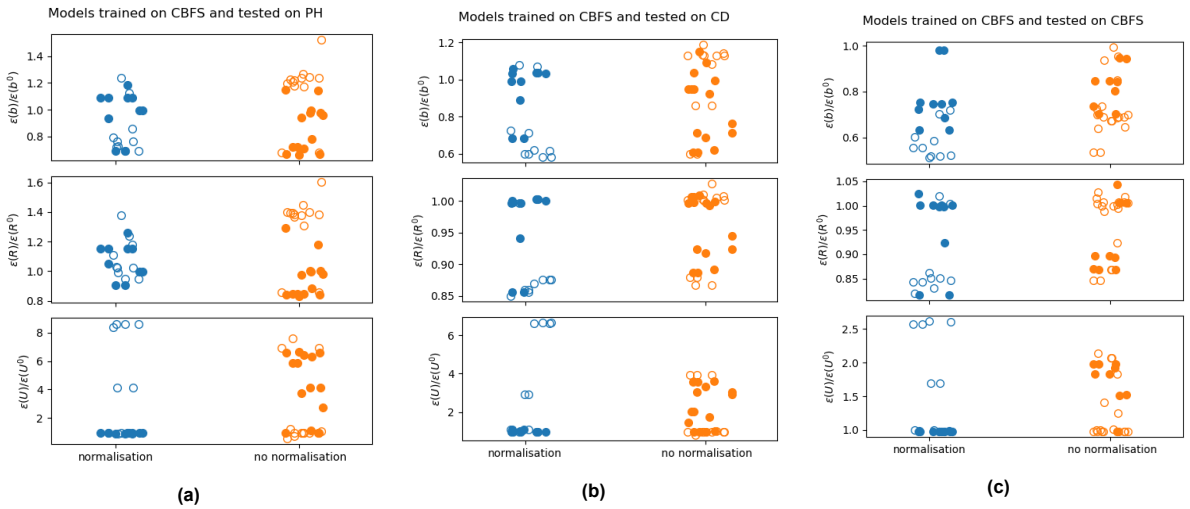


Figure C.21: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the CBFS case using a normalised or not normalised library and tested on the PH, CD and CBFS.

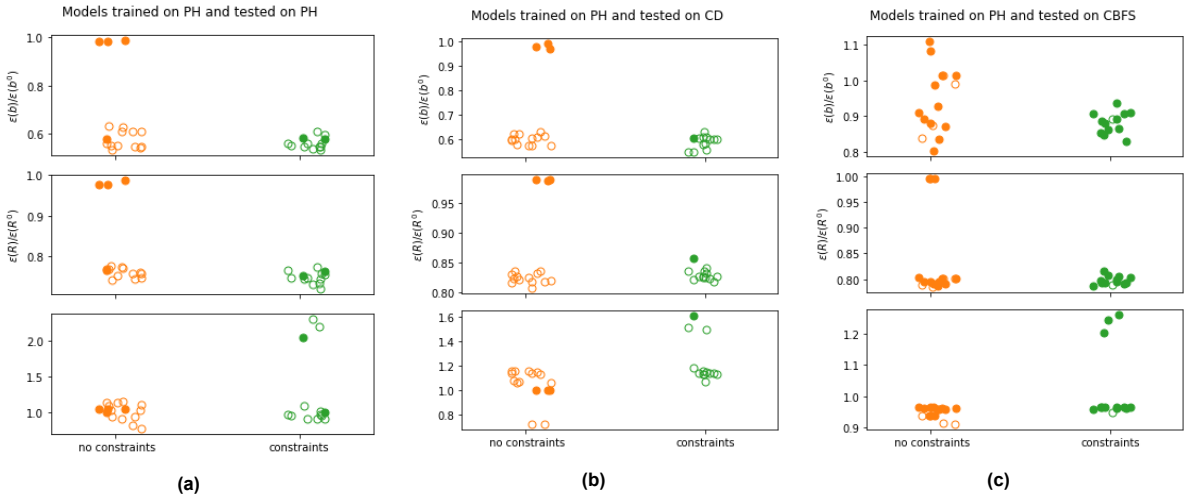


Figure C.22: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the PH case with and without additional constraints, tested on the PH, CD and CBFS case.

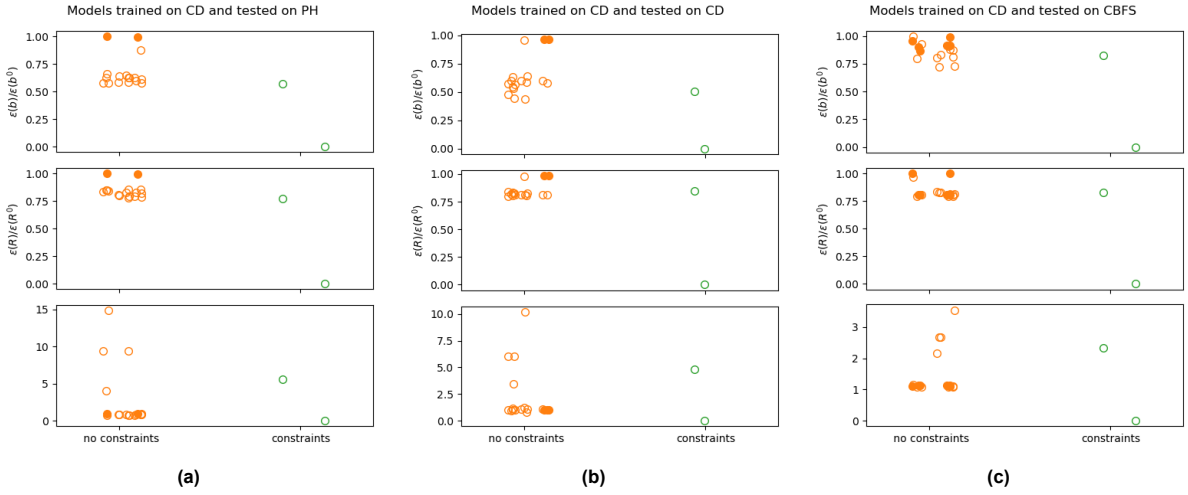


Figure C.23: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the CD case with and without additional constraints, tested on the PH, CD and CBFS case.

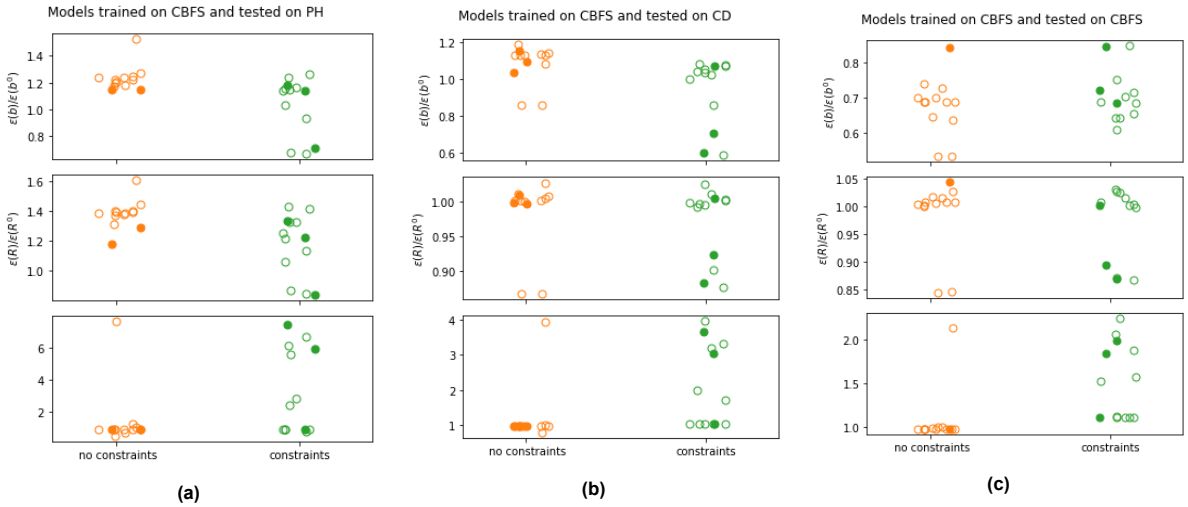


Figure C.24: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the CBFS case with and without additional constraints, tested on the PH, CD and CBFS case.

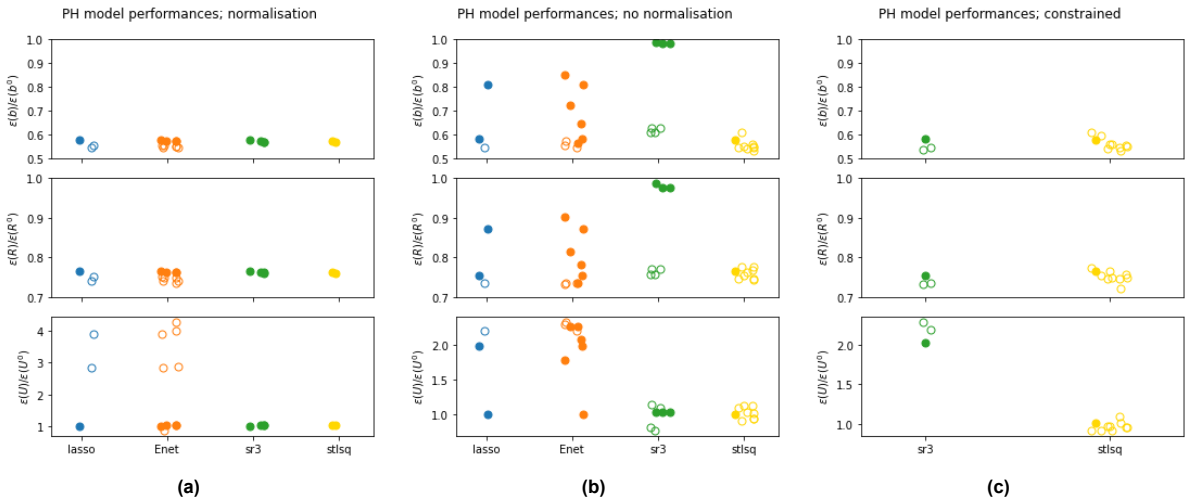


Figure C.25: Comparison of the MSE of predicted b_{ij}, R_{ij} and U of all models discovered and tested on the PH by one of the four regression problems.

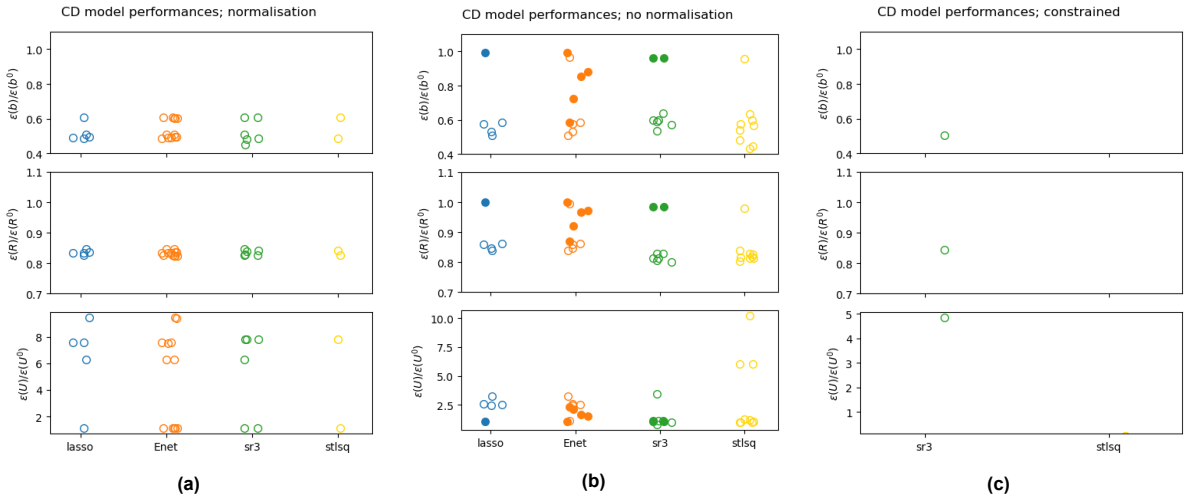


Figure C.26: Comparison of the MSE of predicted b_{ij}, R_{ij} and U of all models discovered and tested on the CD by one of the four regression problems.

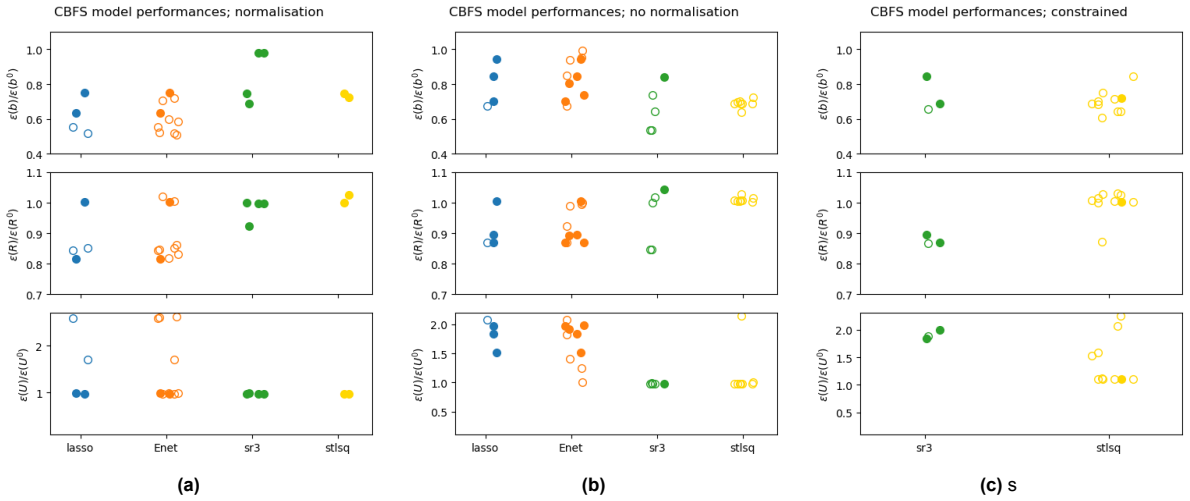


Figure C.27: Comparison of the MSE of predicted b_{ij}, R_{ij} and U of all models discovered and tested on the CBFS by one of the four regression problems.

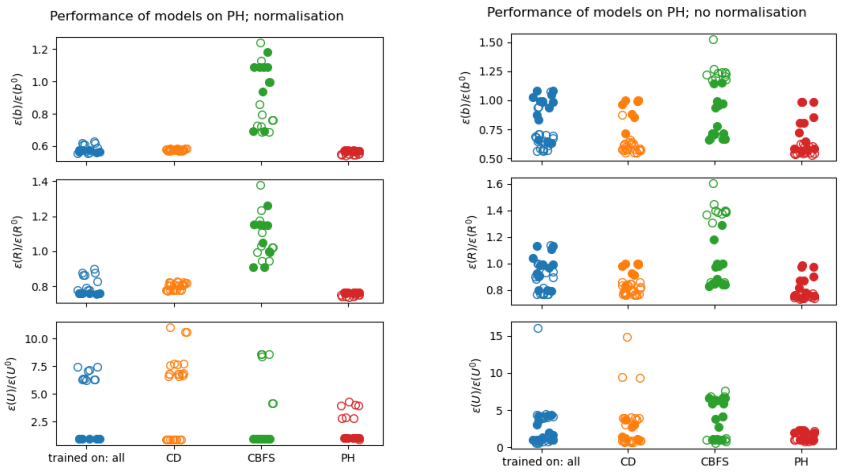


Figure C.28: Comparison of models trained on all cases and trained on a single case, tested on the PH case.

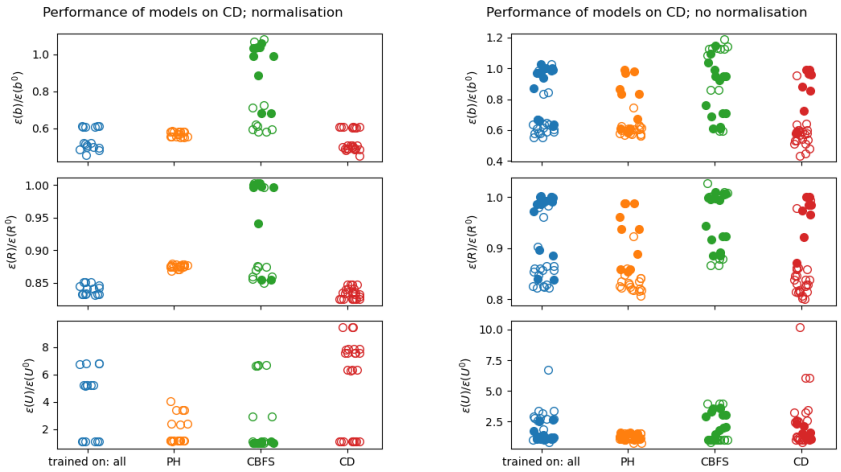


Figure C.29: Comparison of models trained on all cases and trained on a single case, tested on the CD case.

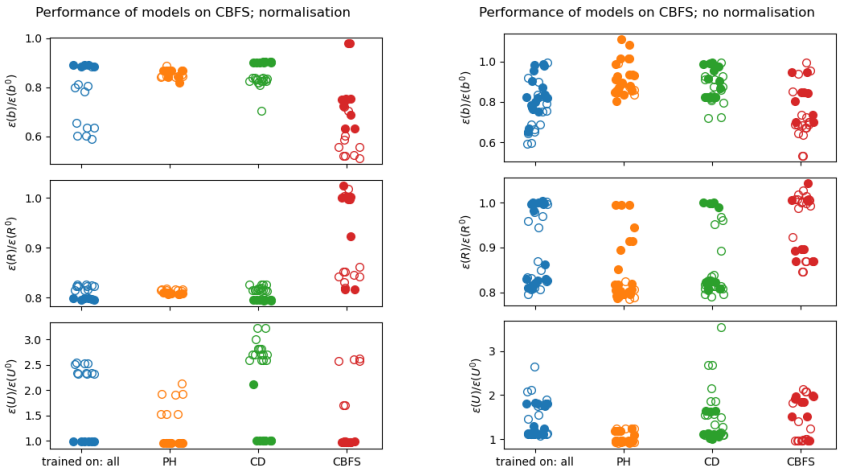


Figure C.30: Comparison of models trained on all cases and trained on a single case, tested on the CBFS case.

D

Discovered Models Invariant Library

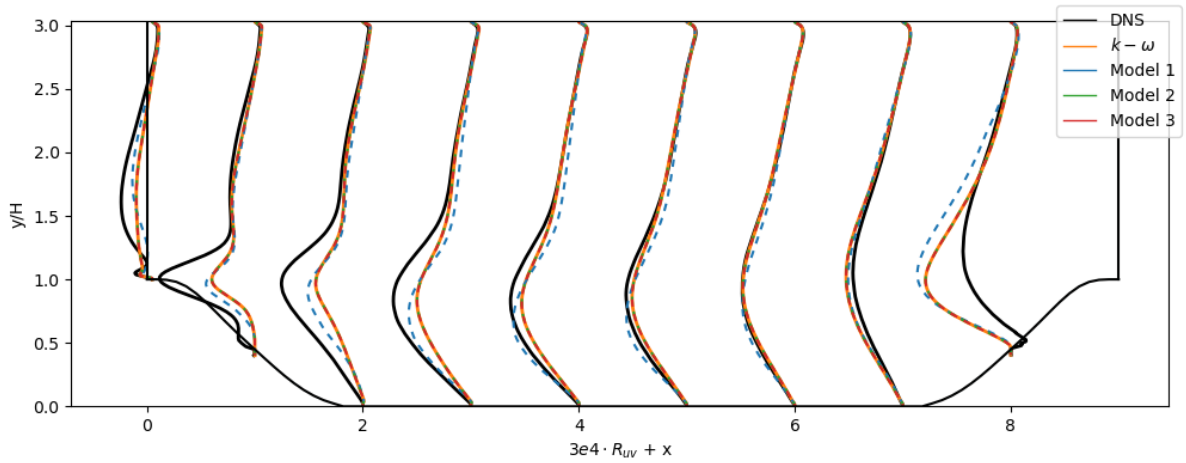
D.1. Best Models

Table D.1: Best performing models for the PH, CD and CBFS case discovered using the normalised invariant library and the normalised mean squared errors of the predicted streamwise velocity for each case.

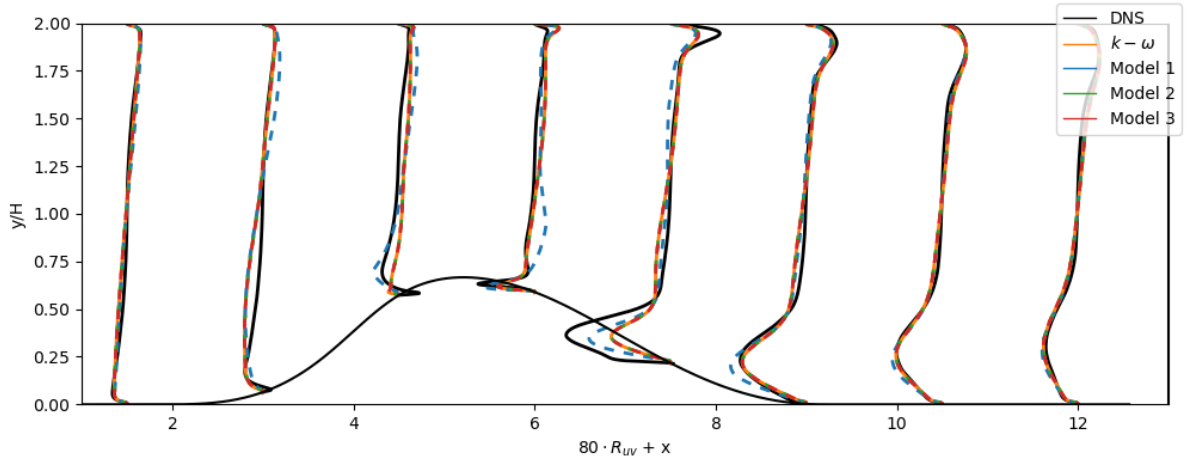
Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = 0.48T_2I_2 - 4.7T_2$	0.8542	1.1915	1.1345
$\mathcal{M}_2 = 0.80T_1I_2^2 + 1.8T_2I_1I_2$	1.1403	1.0284	0.9658
$\mathcal{M}_3 = 0.80T_1I_2^2 + 1.8T_2I_1I_2$	1.1403	1.0284	0.9658

Table D.2: Best performing models for the PH, CD and CBFS case discovered using the not normalised invariant library and the normalised mean squared errors of the predicted streamwise velocity for each case.

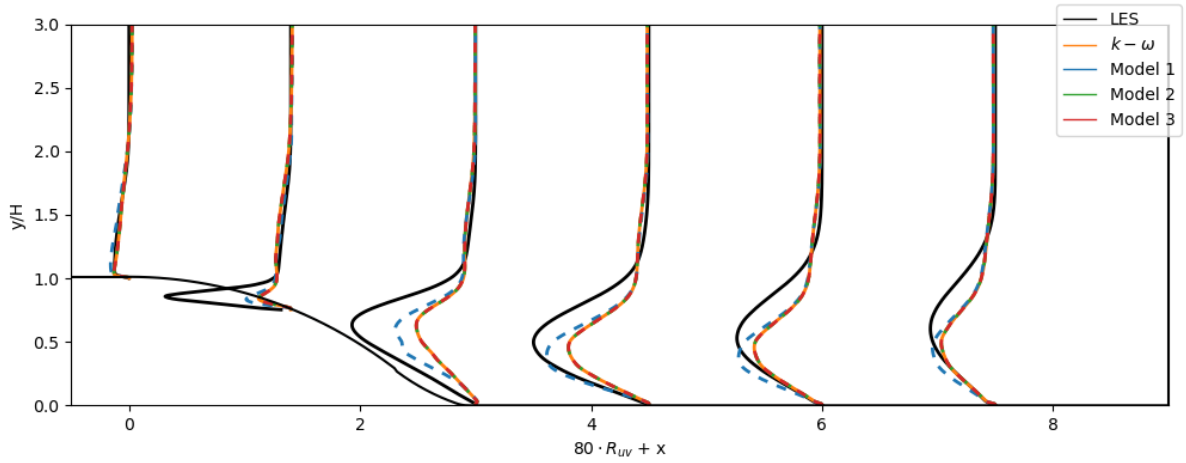
Models	PH $\epsilon(U)/\epsilon(U^0)$	CD $\epsilon(U)/\epsilon(U^0)$	CBFS $\epsilon(U)/\epsilon(U^0)$
$\mathcal{M}_1 = (-31.4I_1 + 38.9I_2)T_2 + (-645.9I_1 + 403.1I_2 + 51.9)T_3$	0.7478	1.0286	1.1269
$\mathcal{M}_2 = (-31.4I_1 + 38.9I_2)T_2 + (-645.9I_1 + 403.1I_2 + 51.9)T_3$	0.7478	1.0286	1.1269
$\mathcal{M}_3 = -1.13T_2$	1.1089	1.0607	0.9693



(a) PH

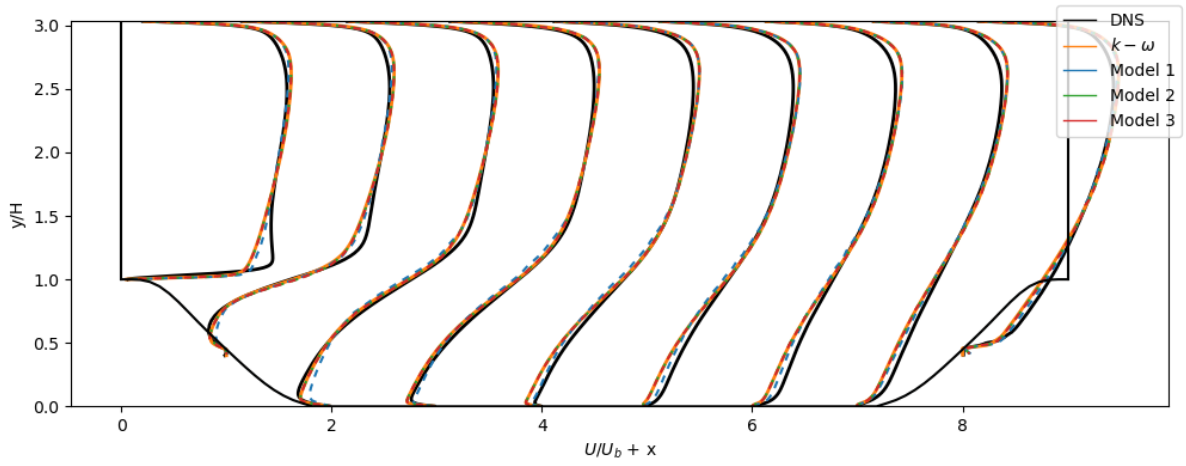


(b) CD

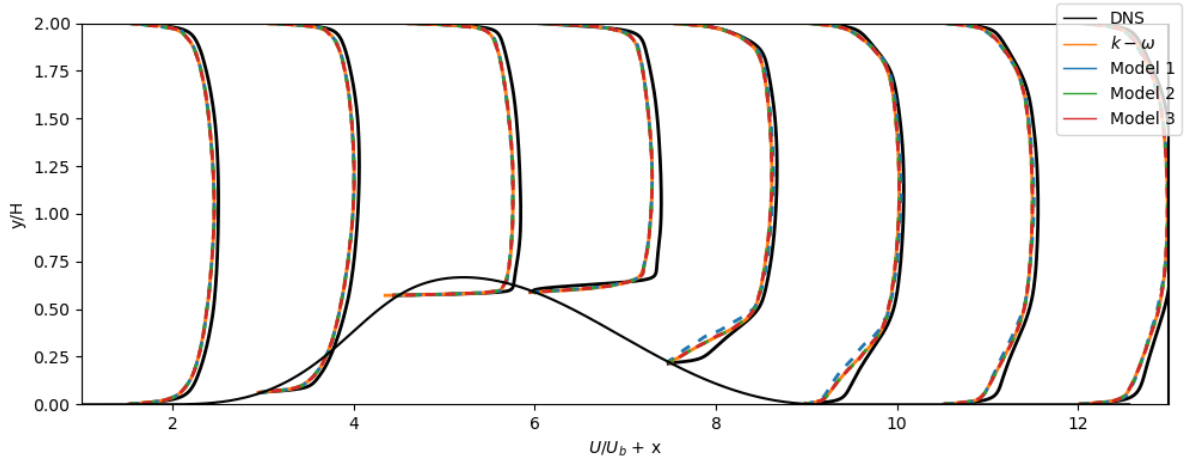


(c) CBFS

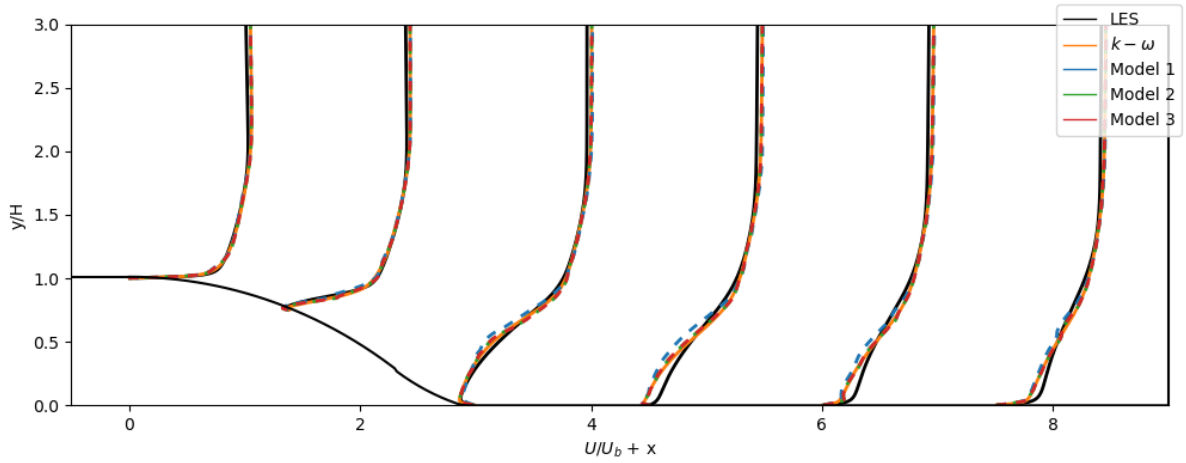
Figure D.1: Shear Reynolds stress of the best performing models (Table D.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

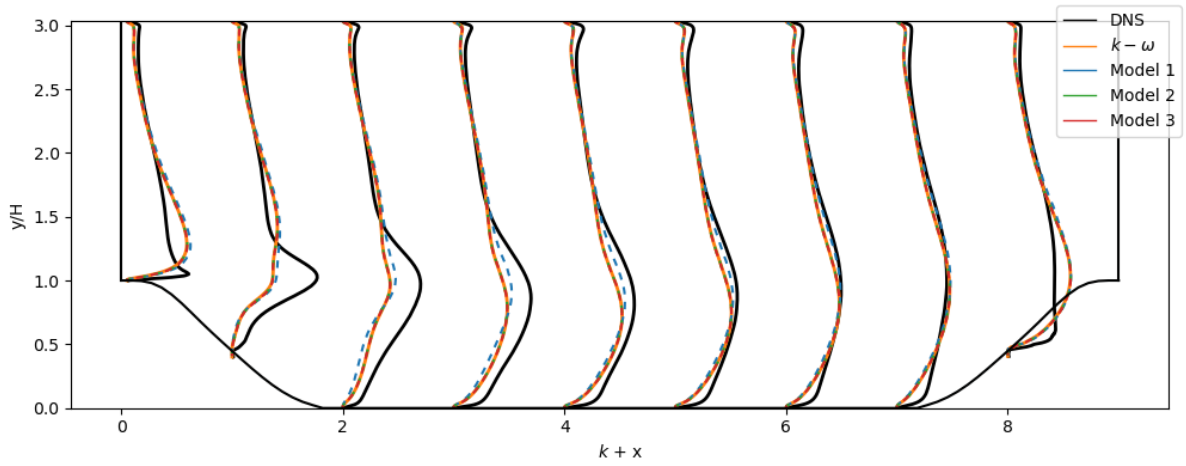


(b) CD

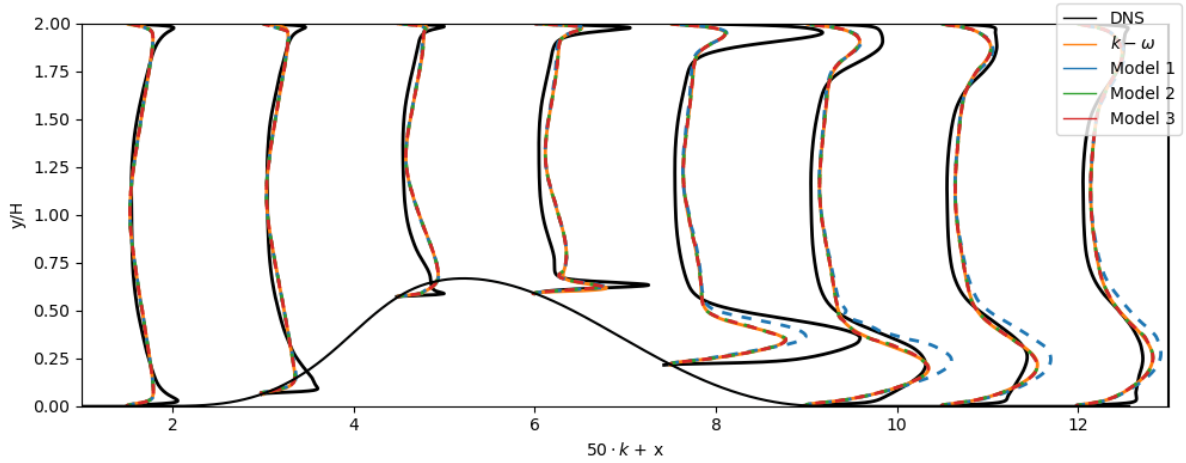


(c) CBFS

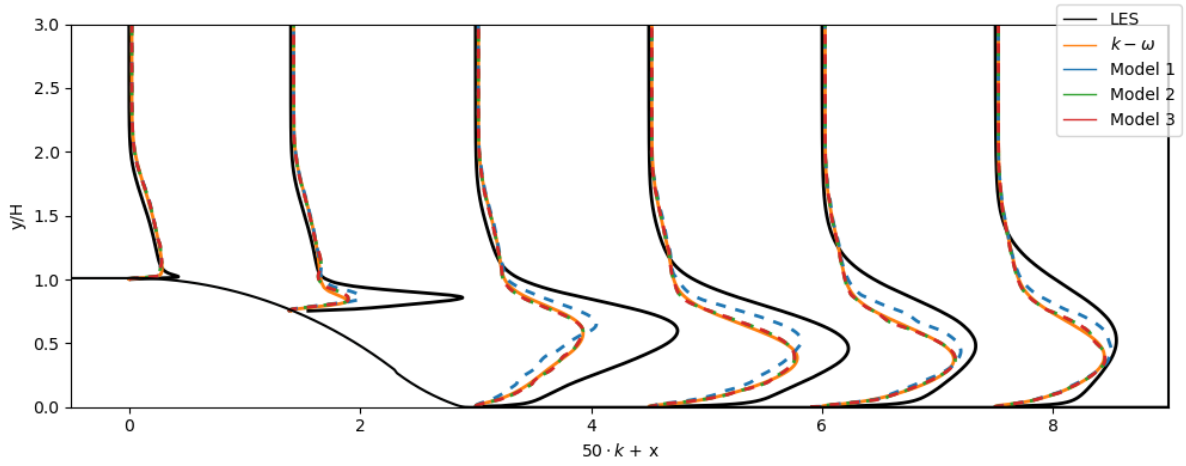
Figure D.2: Streamwise velocity of the propagated best performing models (Table D.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

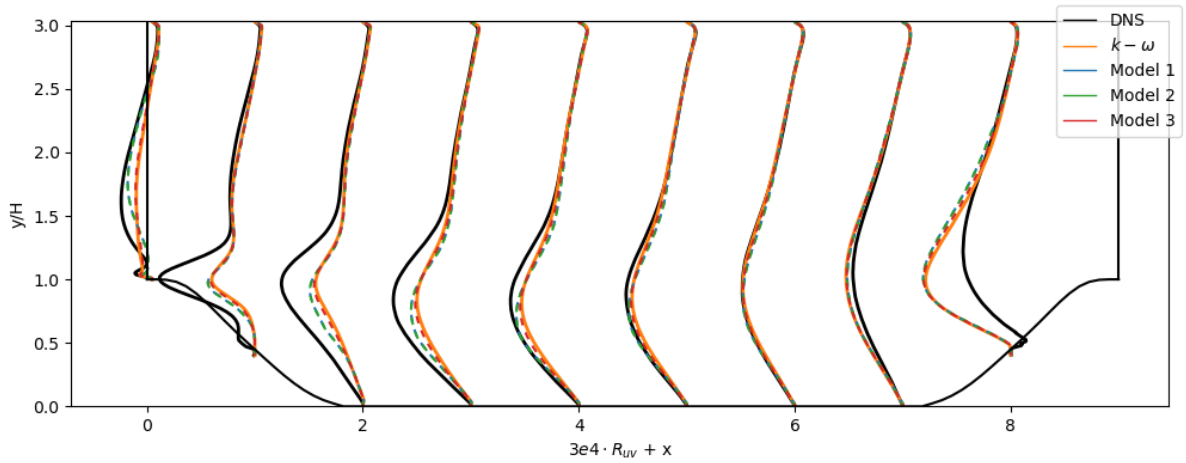


(b) CD

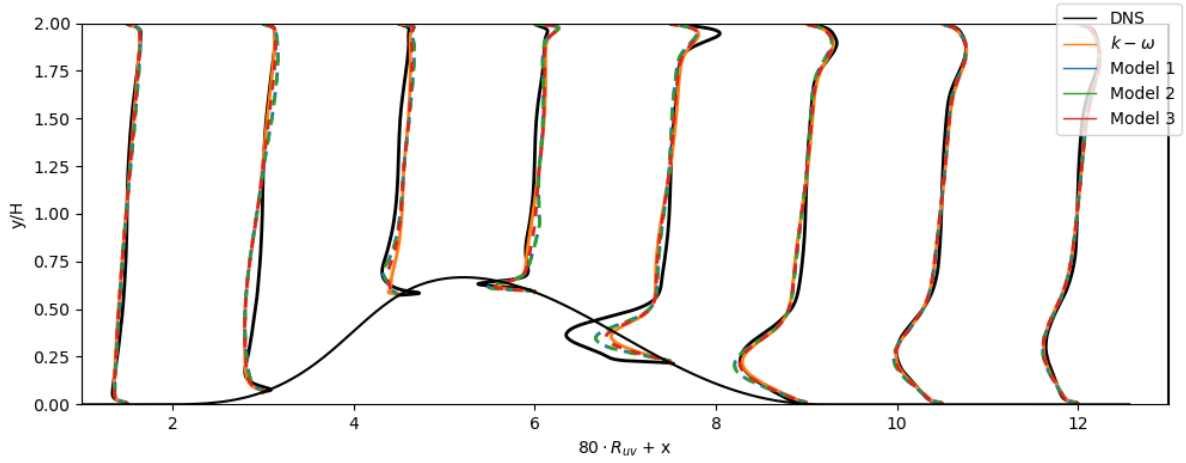


(c) CBFS

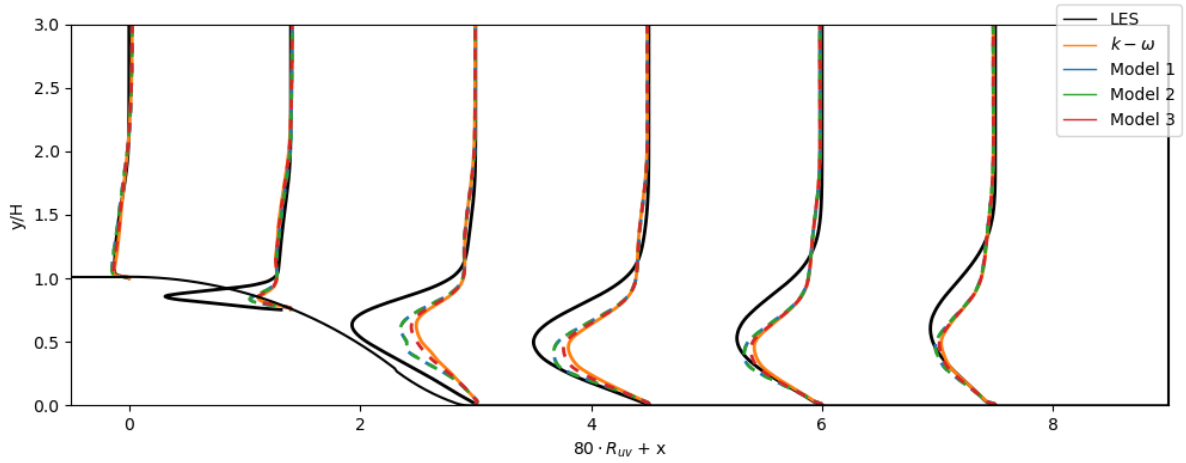
Figure D.3: Kinetic energy of the propagated best performing models (Table D.1) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

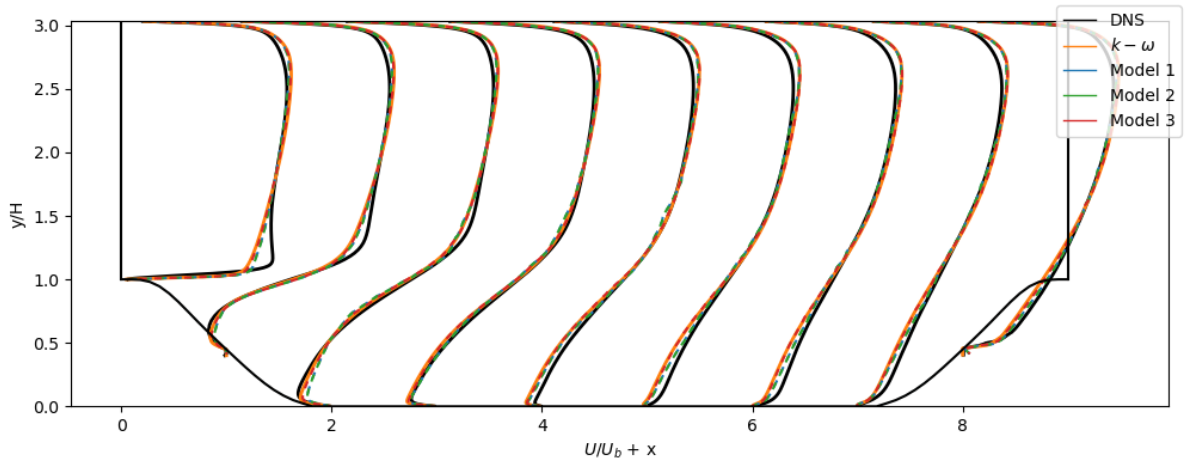


(b) CD

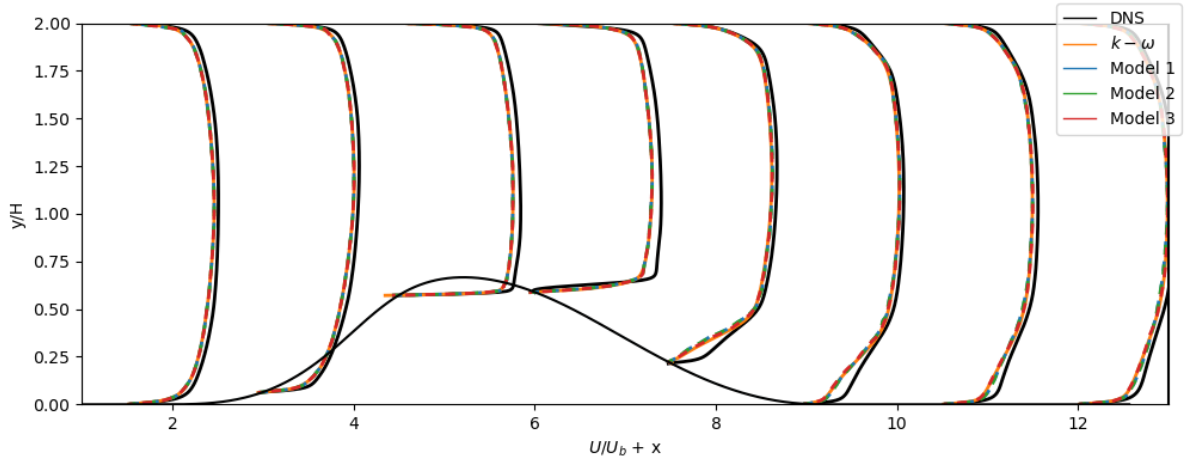


(c) CBFS

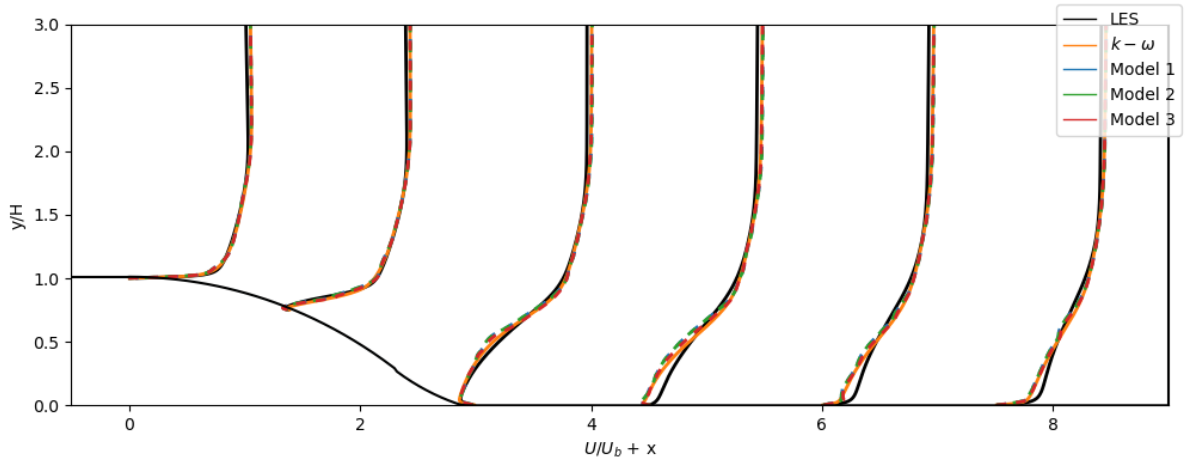
Figure D.4: Shear Reynolds stress of the best performing models (Table D.2) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH

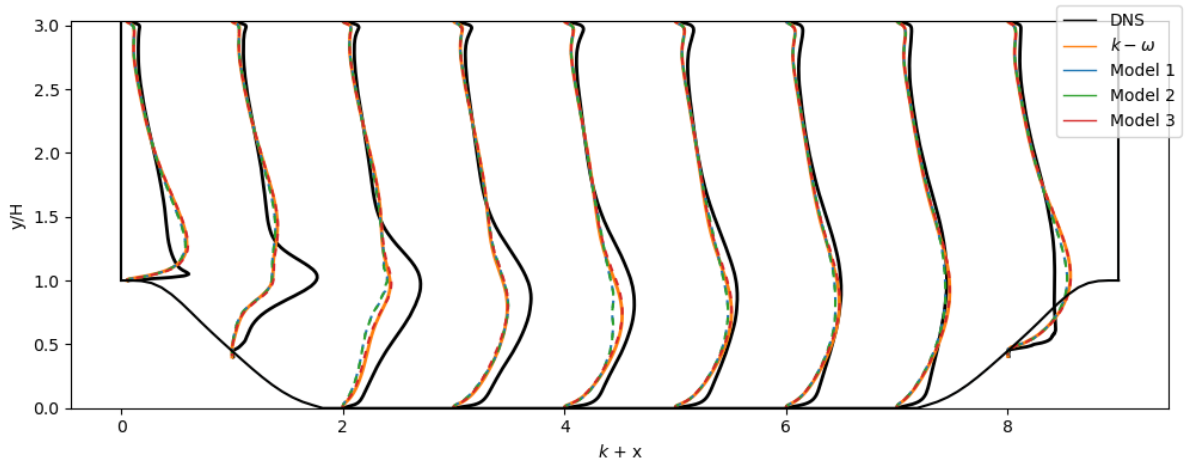


(b) CD

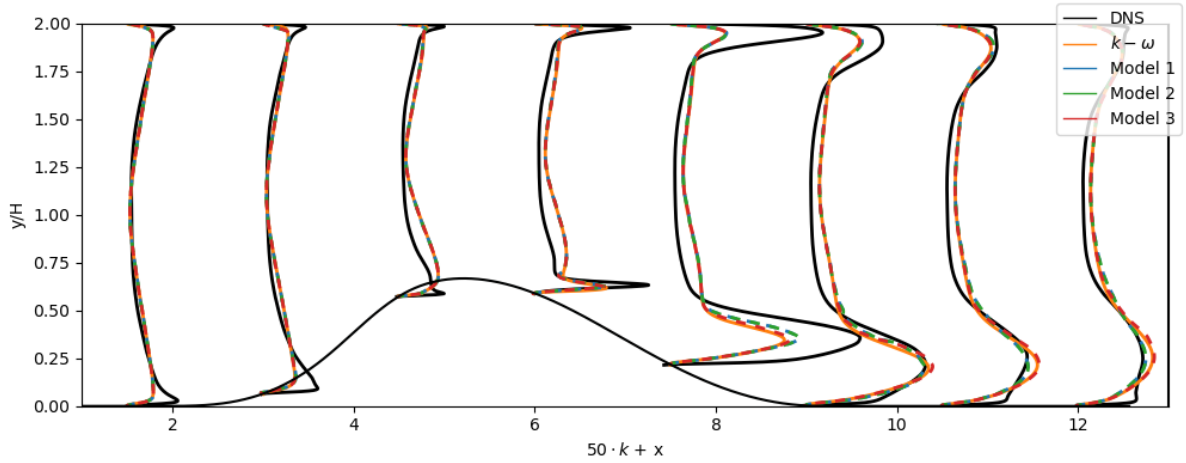


(c) CBFS

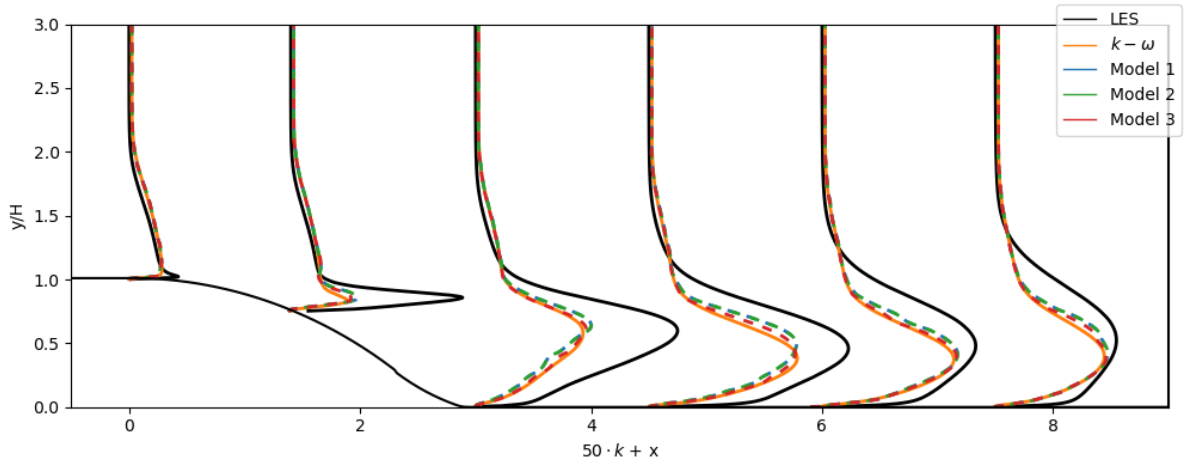
Figure D.5: Streamwise velocity of the propagated best performing models (Table D.2) compared to DNS and $k-\omega$ RANS data for each case.



(a) PH



(b) CD



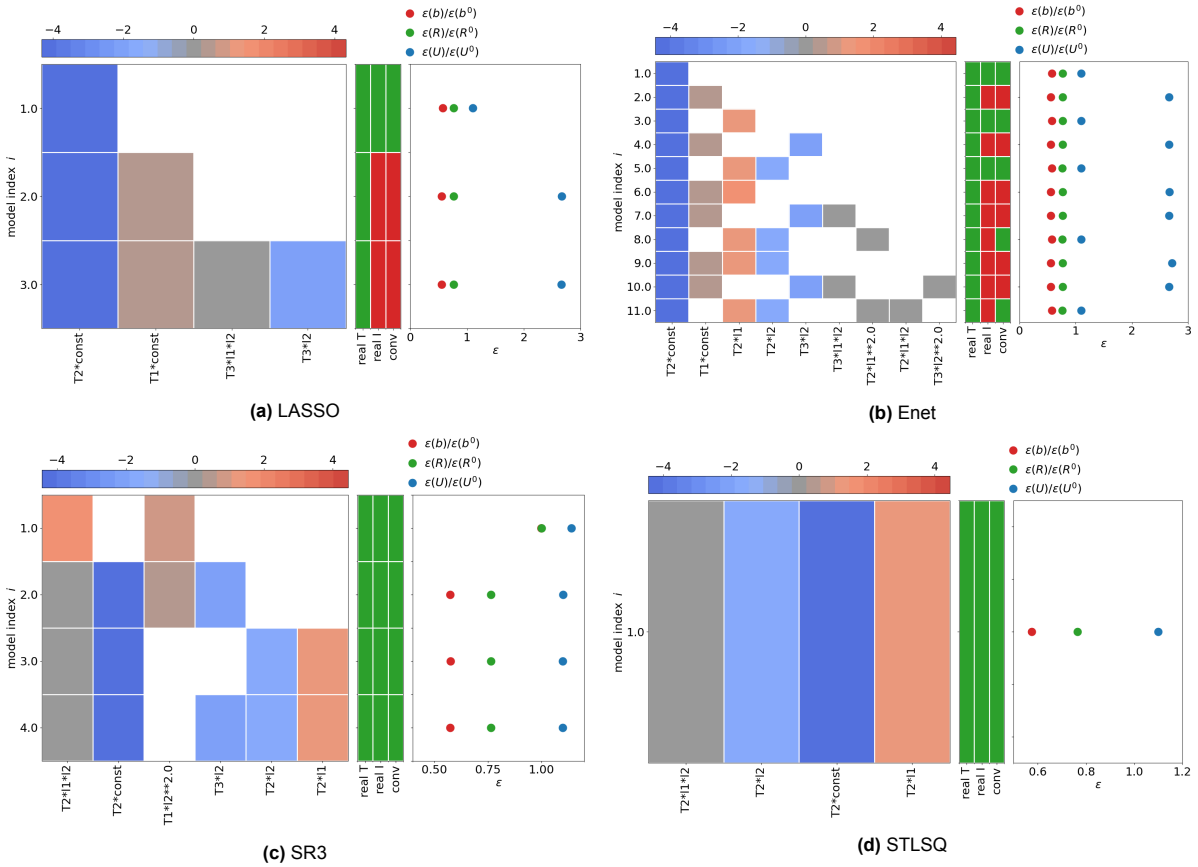
(c) CBFS

Figure D.6: Kinetic energy of the propagated best performing models (Table D.2) compared to DNS and $k-\omega$ RANS data for each case.

D.2. Discovered Models PH

Figure D.7: Feature selection per regression problem and setting for the PH case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

Discovered Models with Normalisation



(a) LASSO

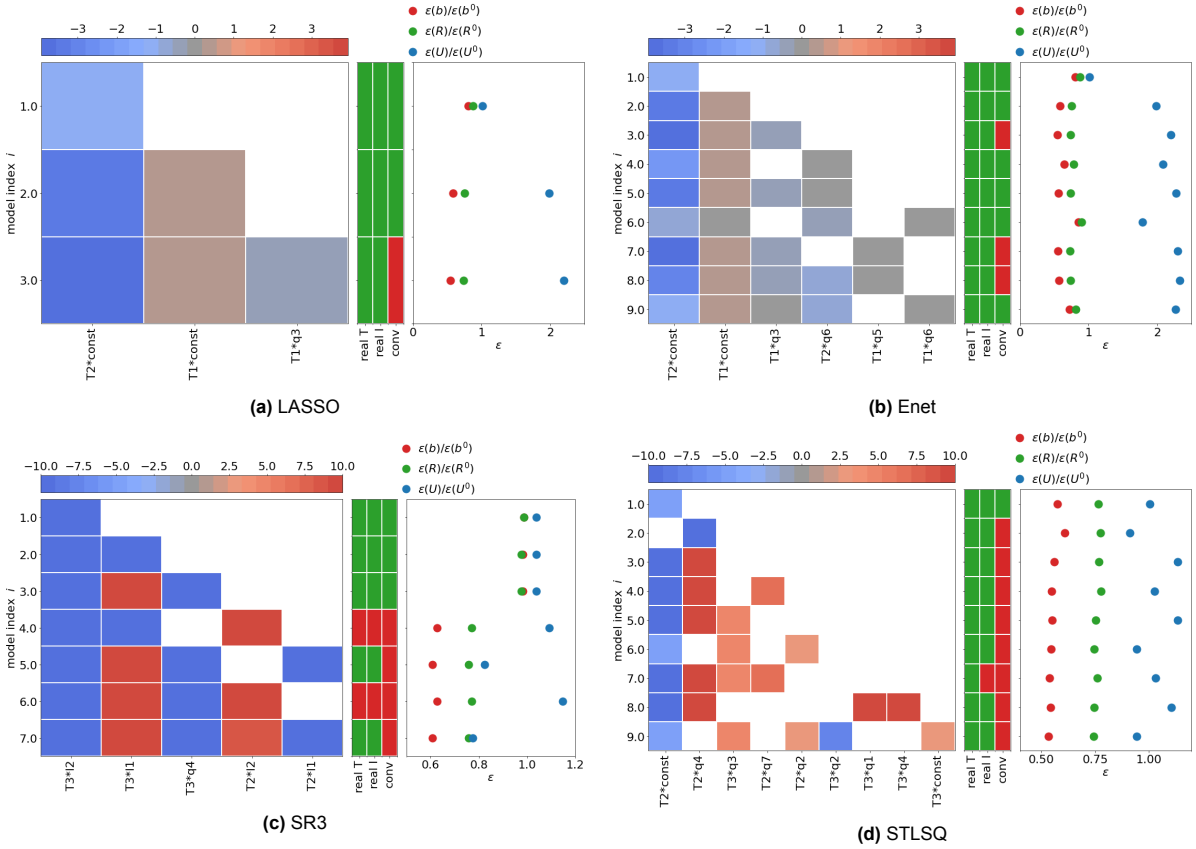
(b) Enet

(c) SR3

(d) STLSQ

Figure D.8: Discovered models by each regressor on the PH case using the normalised library.

Discovered Models without Normalisation

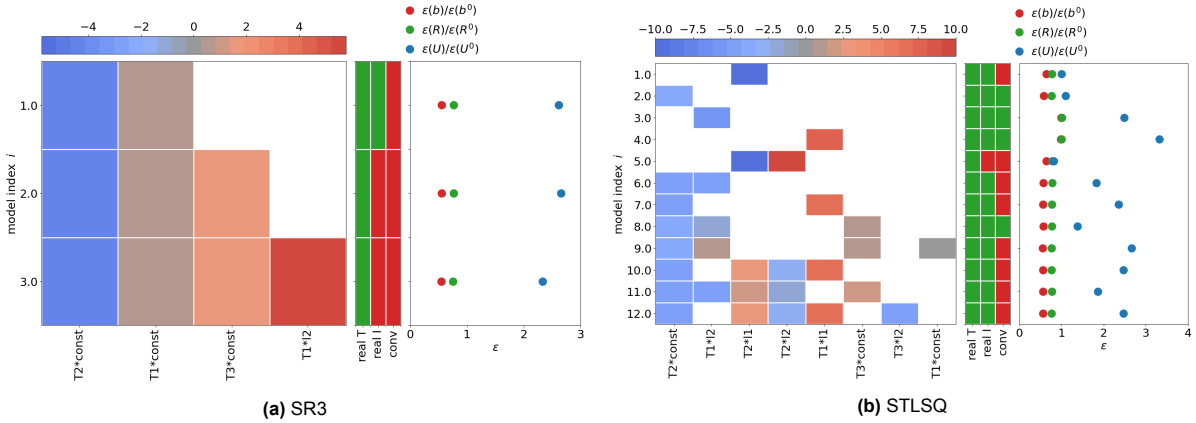


(a) LASSO **(b)** Enet

(c) SR3 **(d)** STLSQ

Figure D.9: Discovered models by each regressor on the PH case using the not normalised library.

Discovered Models with Constraints



(a) SR3 **(b)** STLSQ

Figure D.10: Discovered models by each constrained regressor on the PH case using the not normalised library.

D.3. Discovered Models CD

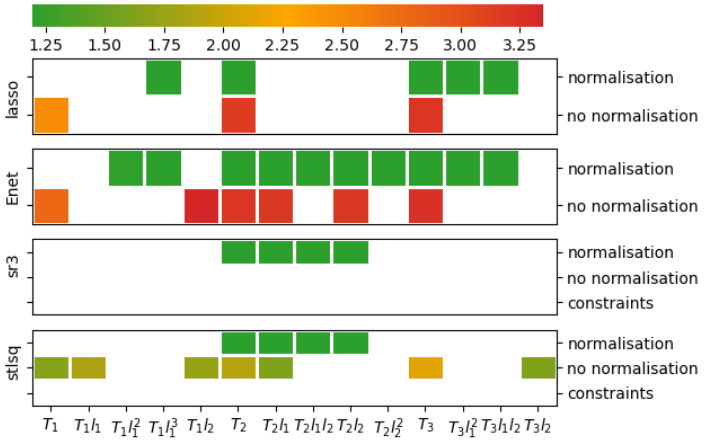
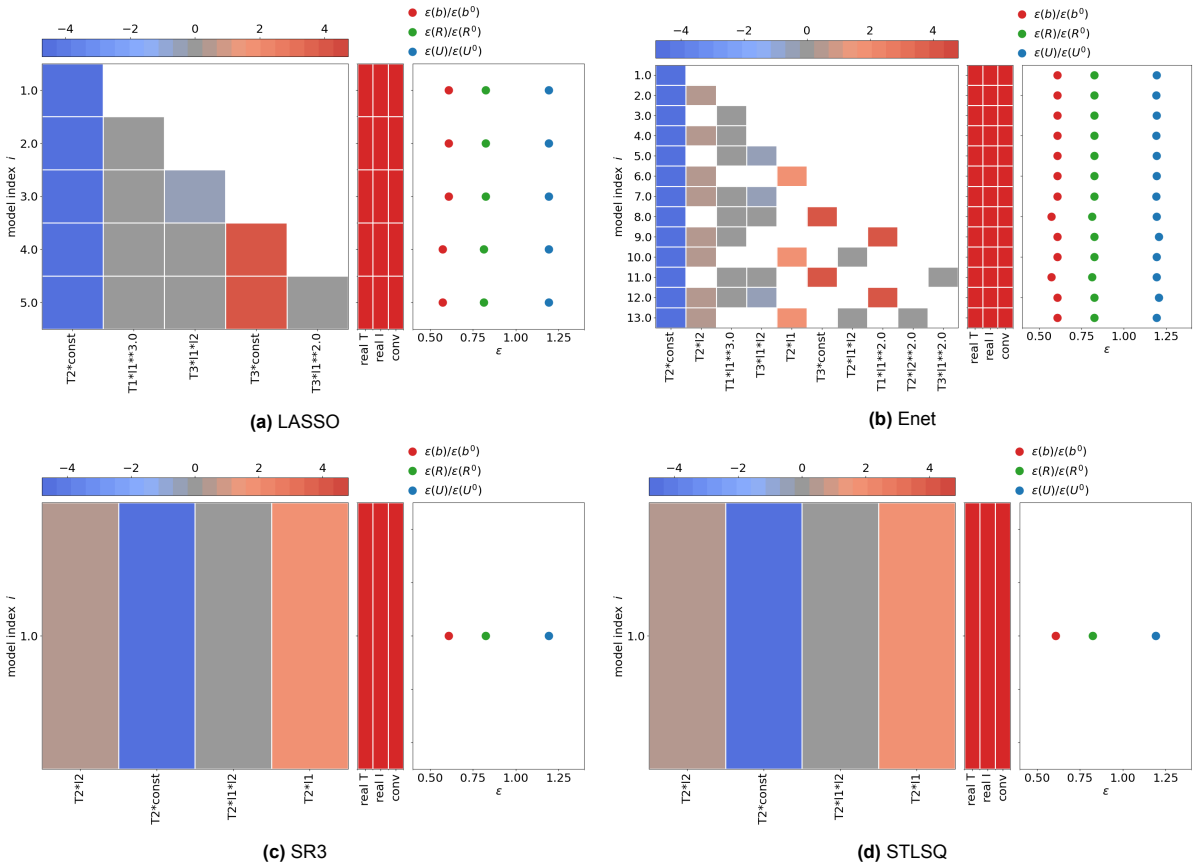


Figure D.11: Feature selection per regression problem and setting for the CD case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

Discovered Models with Normalisation

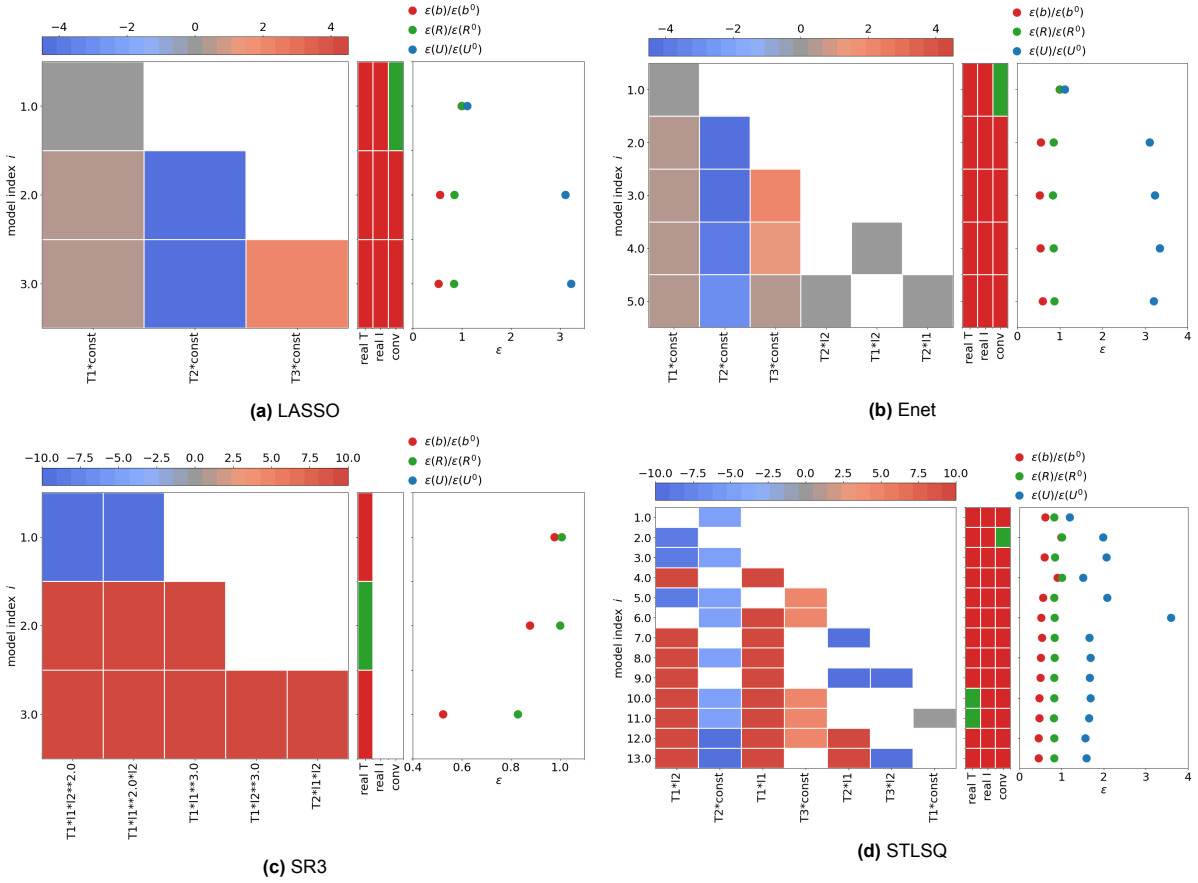


(a) LASSO **(b)** Enet

(c) SR3 **(d)** STLSQ

Figure D.12: Discovered models by each regressor on the CD case using the normalised library.

Discovered Models without Normalisation



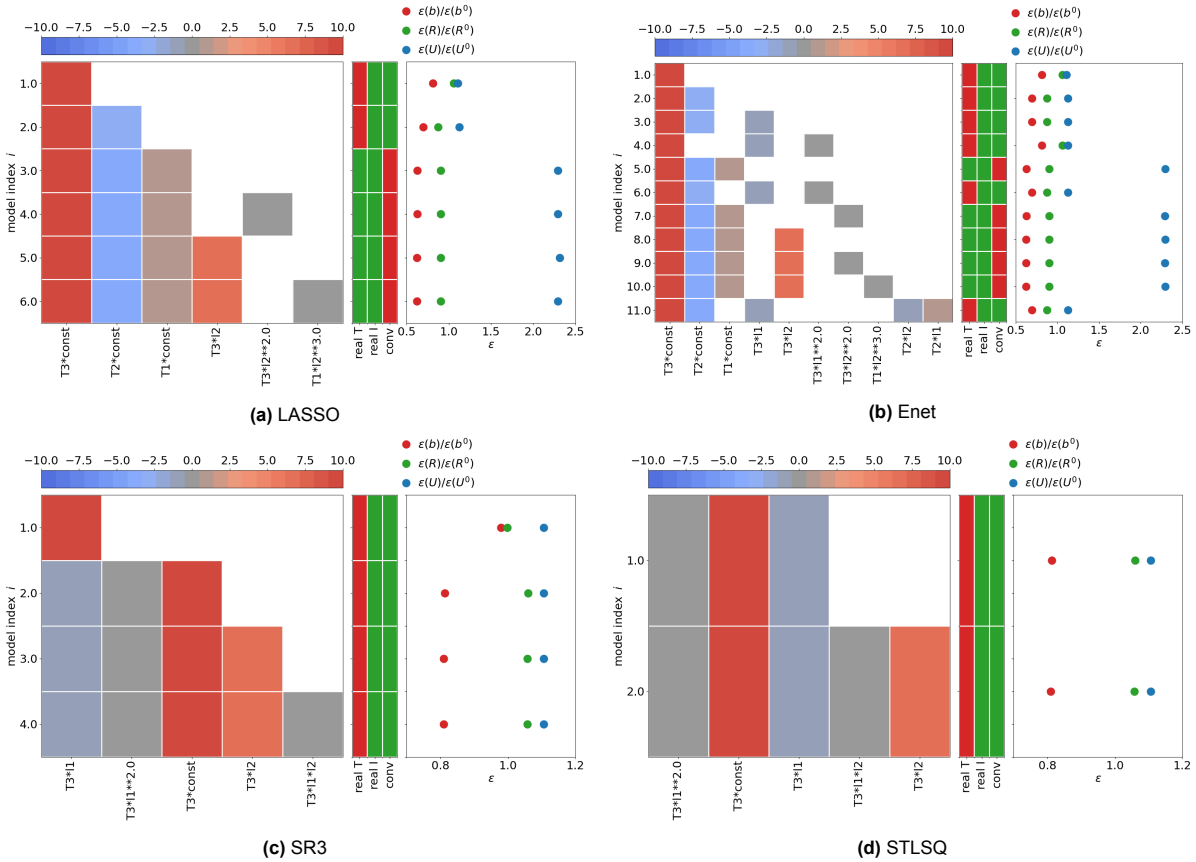
(a) LASSO **(b)** Enet **(c)** SR3 **(d)** STLSQ

Figure D.13: Discovered models by each regressor on the CD case using the not normalised library.

D.4. Discovered Models CBFS

Figure D.14: Feature selection per regression problem and setting for the CBFS case. The colours indicate the mean of the errors of the velocity predictions of the models which contain that function.

Discovered Models with Normalisation



(a) LASSO

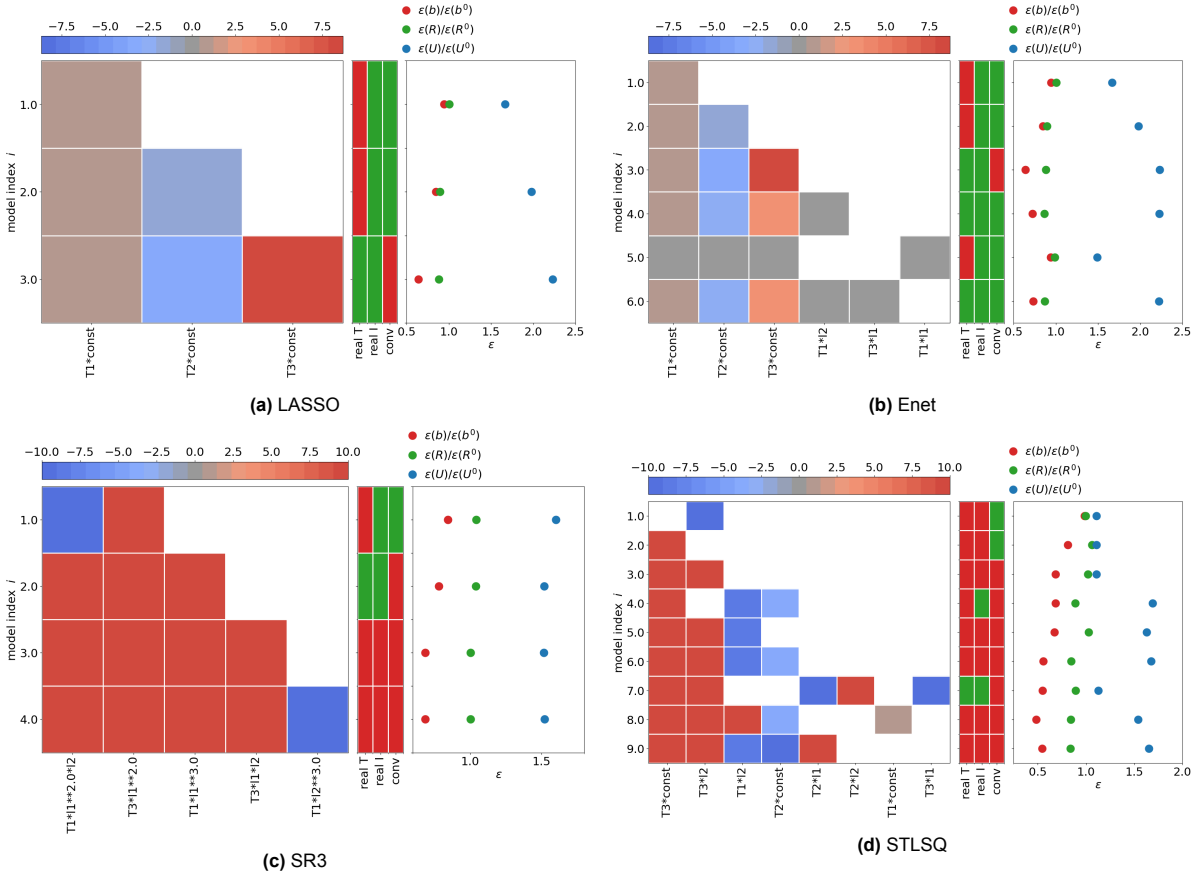
(b) Enet

(c) SR3

(d) STLSQ

Figure D.15: Discovered models by each regressor on the CBFS case using the normalised library.

Discovered Models without Normalisation

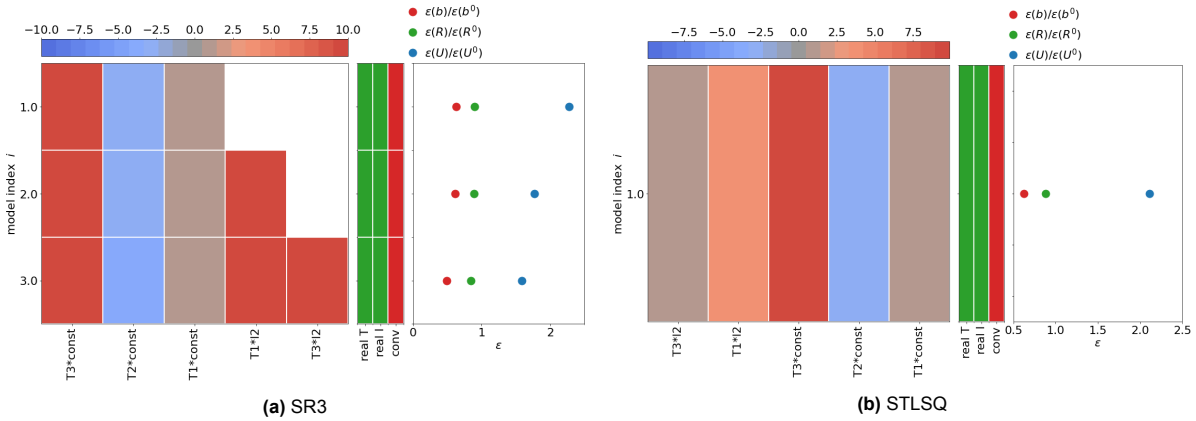


(a) LASSO **(b)** Enet

(c) SR3 **(d)** STLSQ

Figure D.16: Discovered models by each regressor on the CBFS case using the not normalised library.

Discovered Models with Constraints



(a) SR3 **(b)** STLSQ

Figure D.17: Discovered models by each constrained regressor on the CBFS case using the not normalised library.

D.5. Performances of Discovered Models

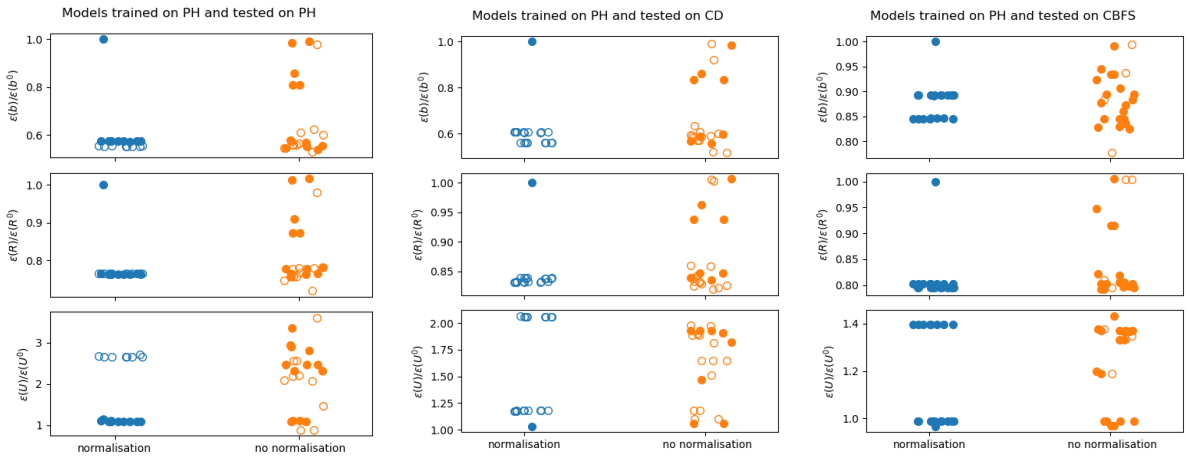


Figure D.18: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the PH case using a normalised or not normalised library and tested on the PH, CD and CBFS.

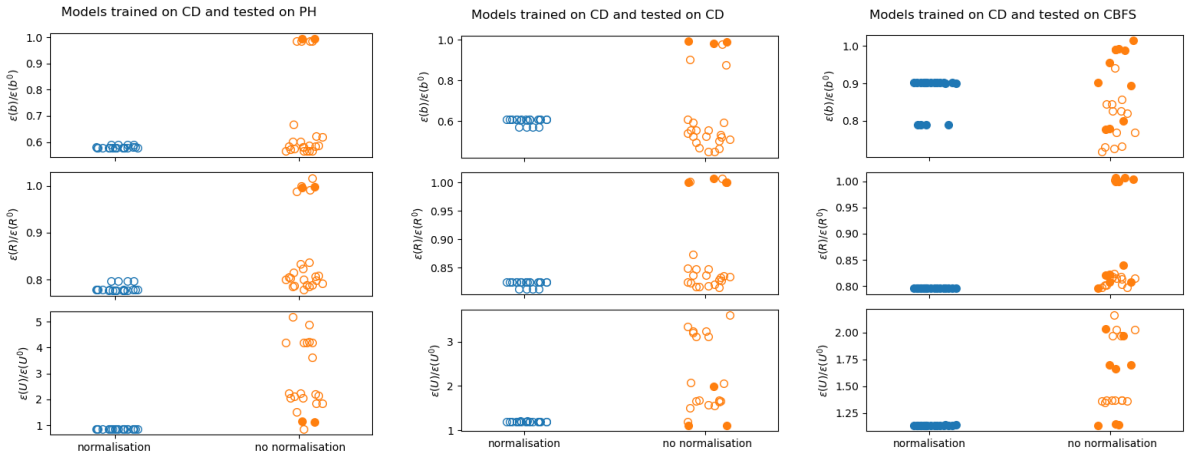


Figure D.19: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the CD case using a normalised or not normalised library and tested on the PH, CD and CBFS.

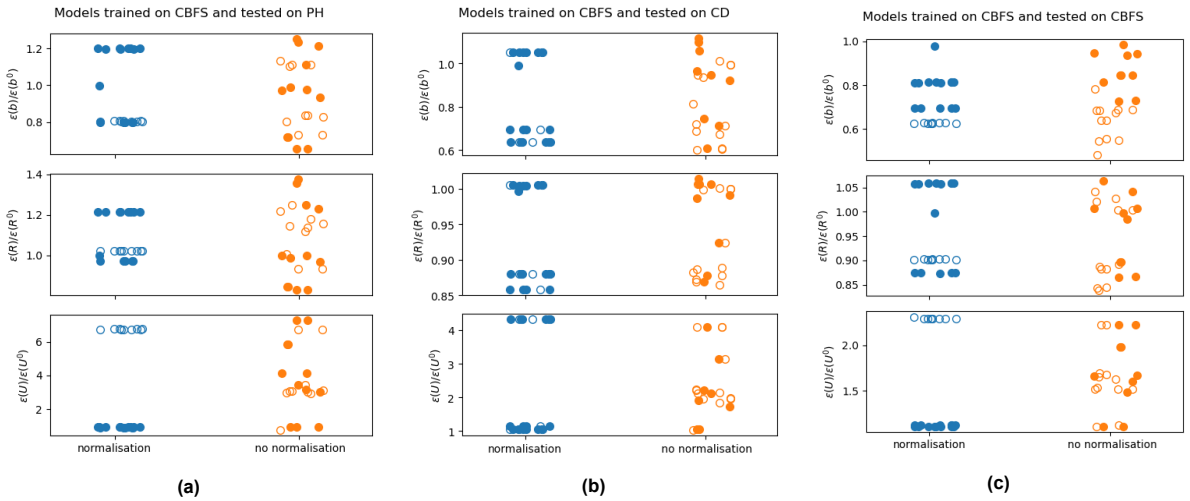


Figure D.20: Comparison of MSE of predicted b_{ij} , R_{ij} and U of all models discovered on the CBFS case using a normalised or not normalised library and tested on the PH, CD and CBFS.

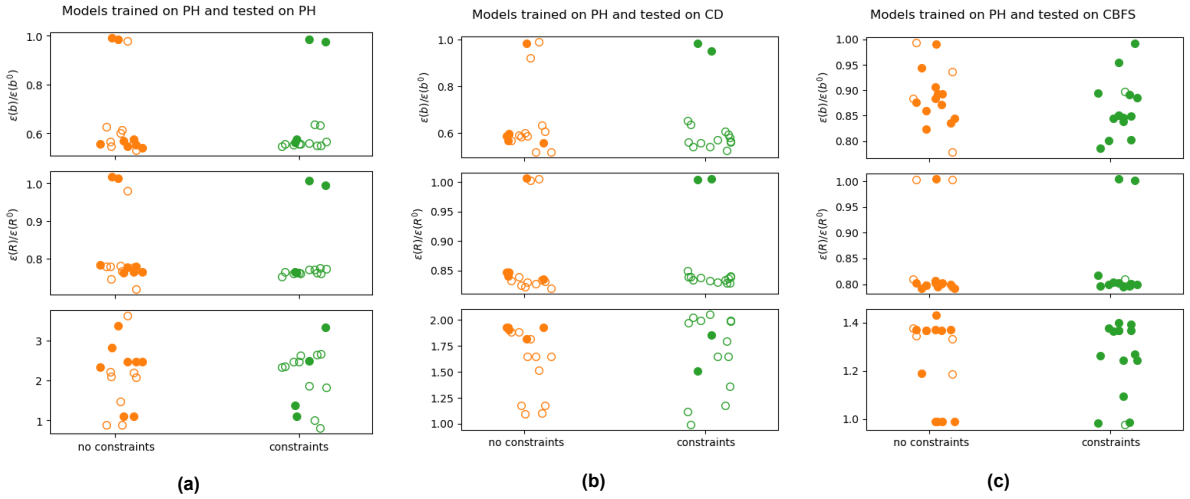


Figure D.21: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the PH case with and without additional constraints, tested on the PH, CD and CBFS case.

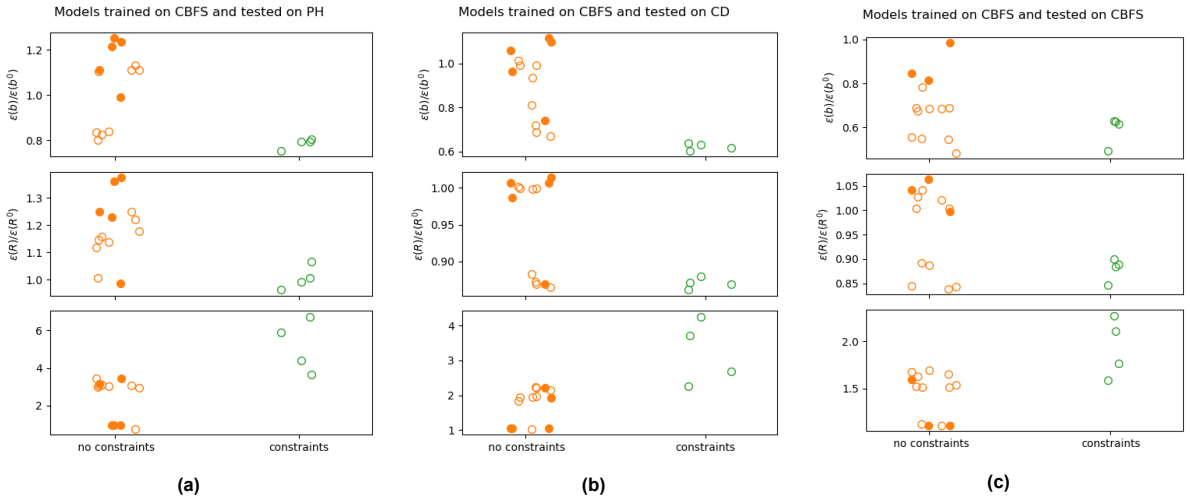


Figure D.22: Comparison of MSE of predicted b_{ij}, R_{ij} and U between models discovered on the CBFS case with and without additional constraints, tested on the PH, CD and CBFS case.

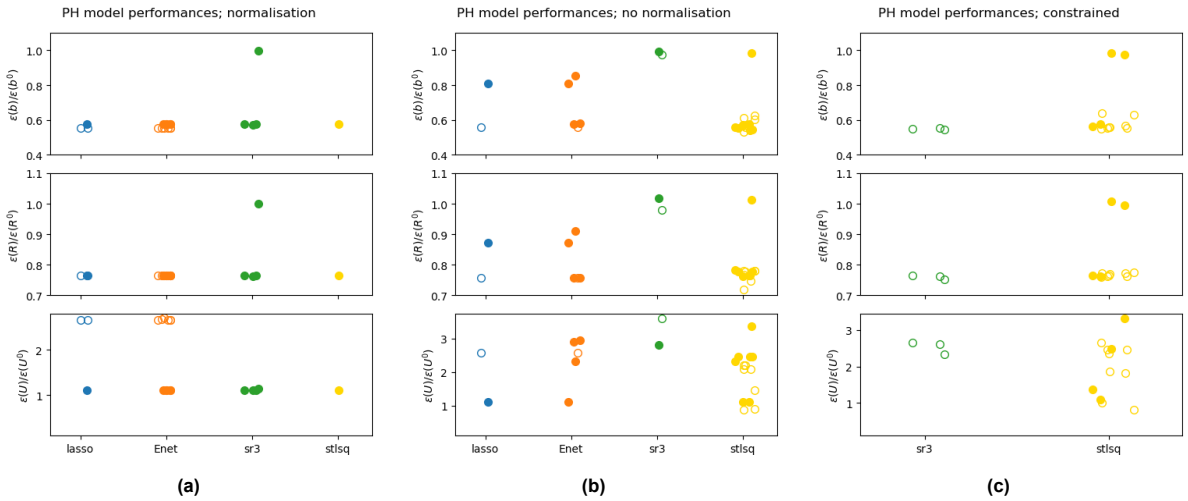


Figure D.23: Comparison of the MSE of predicted b_{ij} , R_{ij} and U of all models discovered and tested on the PH by one of the four regression problems.

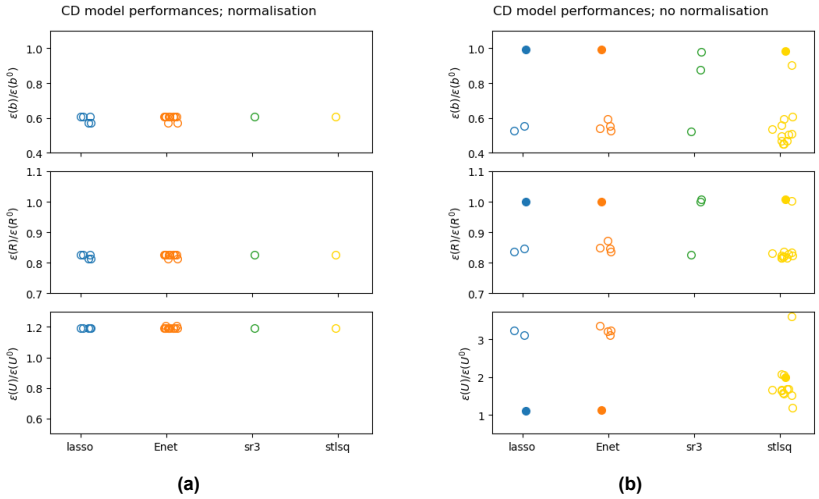


Figure D.24: Comparison of the MSE of predicted b_{ij} , R_{ij} and U of all models discovered and tested on the CD by one of the four regression problems.

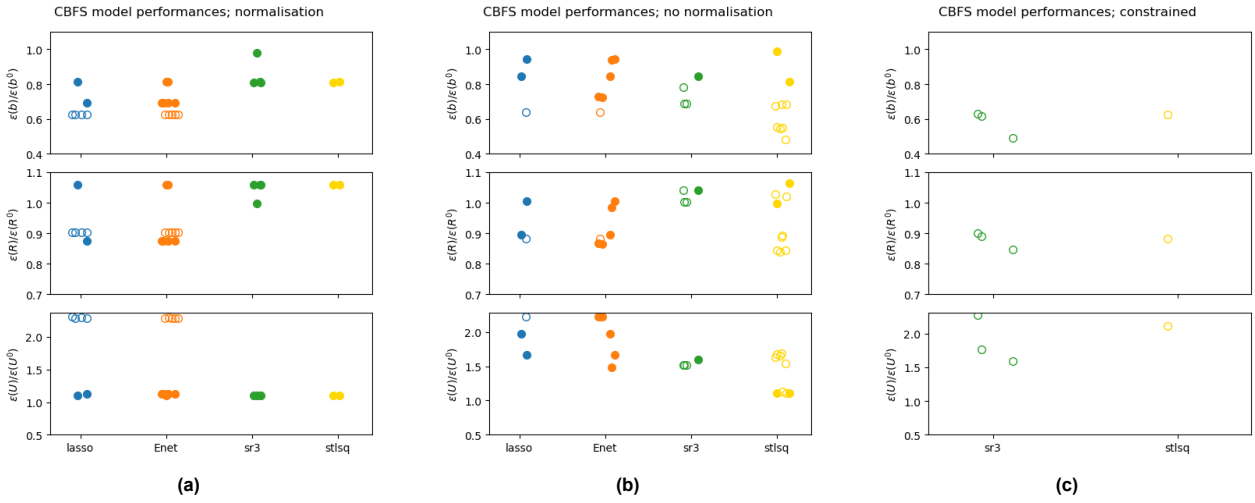


Figure D.25: Comparison of the MSE of predicted b_{ij} , R_{ij} and U of all models discovered and tested on the CBFS by one of the four regression problems.

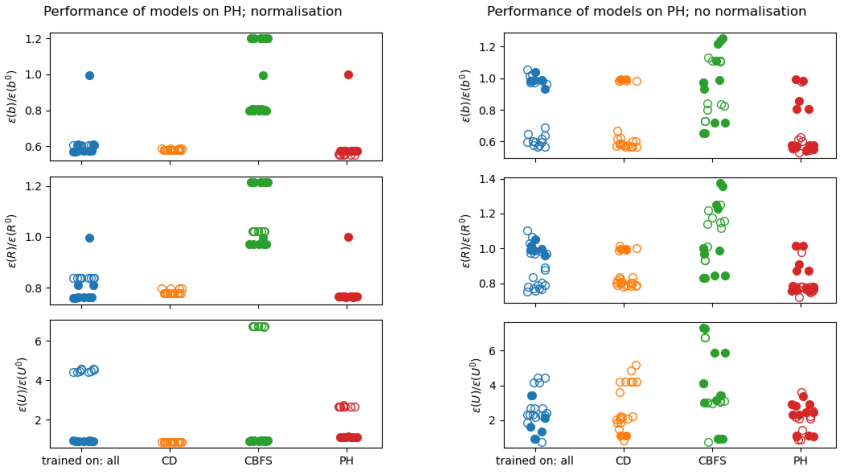


Figure D.26: Comparison of models trained on all cases and trained on a single case, tested on the PH case.

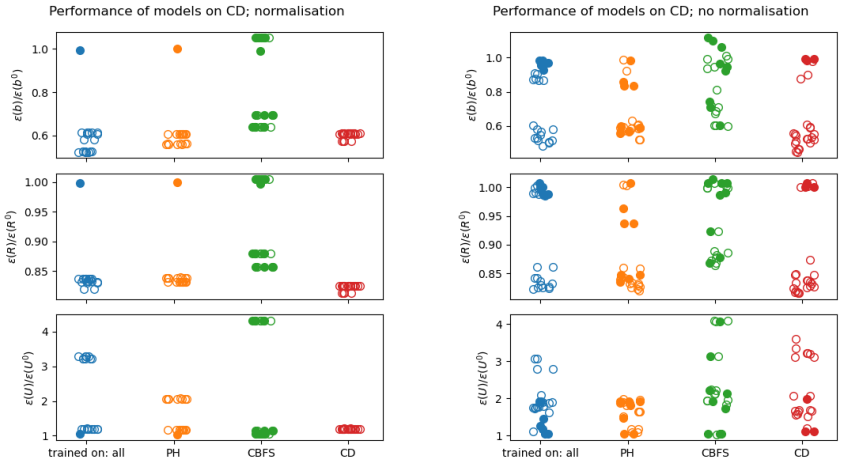


Figure D.27: Comparison of models trained on all cases and trained on a single case, tested on the CD case.

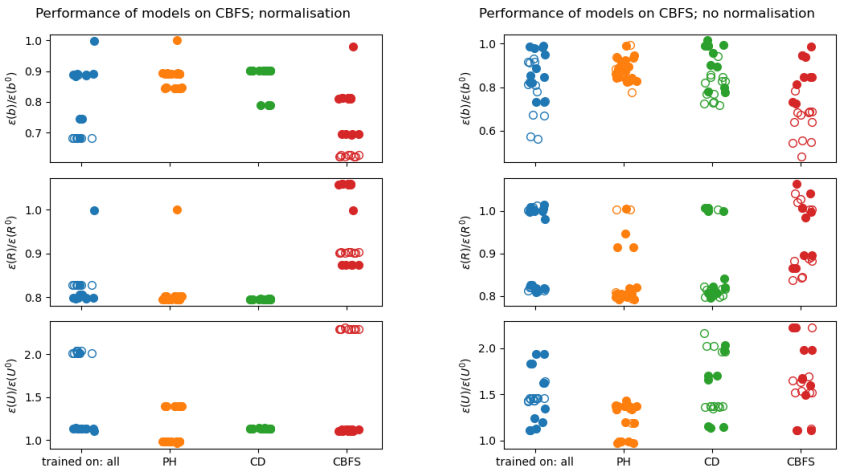


Figure D.28: Comparison of models trained on all cases and trained on a single case, tested on the CBFS case.