Data-Driven Turbulence Modeling

Discovering Turbulence Models with Sparse Symbolic Regression

Elske van Leeuwen

Data-Driven Turbulence Modeling

Discovering Turbulence Models with Sparse Symbolic Regression

by

Elske van Leeuwen

Supervisor Sioux:	Werner Lazeroms
Supervisor TU Delft:	Alexander Heinlein
Exam Committee:	Martin van Gijzen
	Yoeri Dijkstra
Institution:	Delft University of Technology
Place:	Faculty of Applied Mathematics, Delft
Project Duration:	Month, Year - Month, Year

TU Delft

Contents

1 Introduction

1.1. Background

Turbulent flows are ubiquitous in nature and technology, and the simulation of it is an active research topic. Flows around moving objects, such as vehicles and airplanes, are turbulent, and accurate predictions of the turbulent flows are important for the design of that perfect car or airplane having as less drag as possible for example. Also in the design of turbo machines or full wind farms, insight in the turbulent flows is important [24]. Computational Fluid Dynamics (CFD) appears to be an important approach in obtaining this insight on the physical characteristics of turbulence [28]. It relies on governing equations and mathematical models that capture the complexity of turbulent flows, which are solved numerically using computational powers. CFD is favoured compared to experiments as an 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 behavior that is otherwise impossible to gauge. As example, consider a safety analysis of a turbo engine or nuclear reactor to investigate the behavior under sever 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 can be done by the use of the Navier-Stokes equation. Solving this equation directly can be done with direct numerical simulation (DNS). However, in many cases, this is excessively expensive because of the large range of turbulent scales [31]. It can take several months to compute one direct numerical simulation. Therefore, turbulence models are used to perform computationally affordable simulations.These models have the aim to accurately describing the effect of the chaotic behaviour of turbulence on the mean flow. Two strategies for turbulence modeling exists, namely large eddy simulation (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 hence large computational cost. RANS models on the other hand are currently the most widely used models in industry and are expected to remain the norm for simulating turbulent flows, because of their lower computational cost. However, 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 modes, 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 postulates a linear relationship between the Reynolds stresses and the mean strain rate (the symmetric part of the mean-velocity gradient tensor). 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, for example, with the explicit algebraic Reynolds stress model of Walling and Johansson [35], or with the complex Reynolds stress models [10]. However, these nonlinear models are not used often for industrial problems, since they do not give consistent improvement over the LEVM and often have convergence problems [13]. 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 [21].

Recently, there has been an increased interest in using Machine Learning (ML) to improve, augment or develop RANS turbulence models. Over the past decades the computational power has been majorly improved, which contributed to an increase of available high-fidelity data from DNS and fast developments in machine learning techniques. This has led to the introduction of data-driven RANS turbulence modeling. 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.

Many challenges remain in the current state-of-the-art of data-driven RANS models. To begin with the generalizability of data-driven modeling. Ideally, an established data-driven approach can be applied to different or unseen scenario's. However, almost all previous work demonstrated the predictive ability of the model to limited 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 [18]. Furthermore, previous research revealed that an interpretable model is preferred and can enhance the generalization 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 obtain 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 the ill-conditioning of the RANS equations with its closure model [39].

Recently, the relatively new sparse symbolic regression technique [6], has been introduced in data-driven turbulence modeling and the initial results are promising [30], [4]. Sparse symbolic regression generates an algebraic equation to describe the quantity of interest. In terms of intertpretability this is favourable compared to other machine learning techniques, such as Neural Networks (NN) since NNs are black box models. 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. At last, 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, we will apply sparse symbolic regression to find an algebraic closure model for the RANS equations. This will be tested on different geometries and ultimately on a complex industrial problem. One of the goals is to insert more physical knowledge while building the algebraic models to improve generalizability. Furthermore, we aspire to extract physical knowledge while building data-driven models by experimenting with input features and verify if the sparse symbolic regression technique is able to identify the most important features. On numerical level, we analyse the influence of different regression techniques which can be used in the sparse symbolic regression. At last, we would like to get more insight in the performance in terms of generalizability 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 and should be interpretable, generalizable 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 of increasing complexity to train and test the models on, including a complex industrial problem.
- Implement the models in an existing CFD solver (OpenFoam) and assess the improvements in mean-flow over the standard RANS model.
- Assess the generalizability and robustness of the models.

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

Q1 Which features are the most relevant when deriving the algebraic equation?

- 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?

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, generalizability and interpretability?

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

1.3. Structure

In chapter 2, the basics of turbulence and turbulence modeling are explained. Different machine learning techniques, and how they can be applied in turbulence modeling are discussed in chapter 3. chapter 4 contains the methodology of the research project followed by the results in chapter 5. For now, we conclude the report with some intermediate conclusions and a description of the rest of the research in chapter 6.

2 Turbulence and turbulence modeling

This chapter should serve as introduction to turbulence and the two common used simulation strategies, namely RANS and LES. The governing equations will be explained, including the closure problem.

2.1. Turbulent Flows

Turbulence is everywhere around us and it is omnipresent in nature and technology [24]. Examples of turbulent flows in technology are the flows in nozzles and pipes, and flows in devices such as turbo machinery's and heat exchangers. Furthermore, turbulent flows occur almost always around moving objects, such as cars and airplanes. If you once have flied with an airplane you probably experienced turbulence yourself as the powerful shaking of the aircraft. In nature, turbulence can be found on a geophysical scale, for example in our atmosphere or in the oceans. The distribution of air pollution in our atmosphere is mainly controlled by turbulence. Also our weather and even our climate could be seen as a turbulent phenomenon. For many technology problems and applications it is import to determine the effect of turbulence on the performance of that engineering device [9]. In the design of wind turbines, knowledge about the turbulent flow around the blades is essential for an optimal efficiency of the turbine. But also in car and airplane designs, it is demanded to delay the occurrence of turbulence around the boundary surfaces, to decrease the drag and reduce the fuel consumption.

But what is turbulence exactly? Turbulence is a state of fluid motion which is characterised by a chaotic behavior 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 that case the fluid flows smoothly, in parallel layers with no or little mixing. The shift between these two states depends on the ratio between viscous and inertial forces. This ratio is expressed by the Reynolds number. In turbulent flows, the viscous forces are small with respect to the inertial forces resulting in a high Reynolds number.

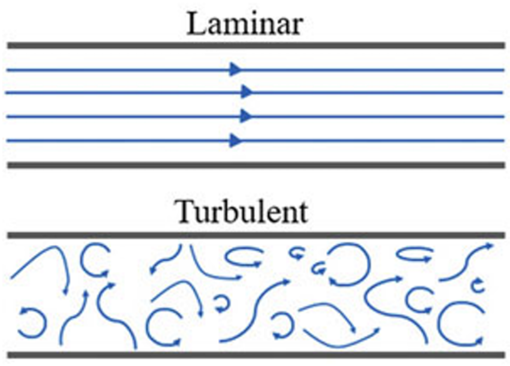


Figure 2.1: Schematic drawing of a laminar and turbulent flow, taken from [28, p. 35]

Turbulence is also 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 motion of a fluid can be described by a complete set of equations, called the Navier-Stokes 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 [24]. The NS equations emanate from the laws of conservation of mass, momentum and energy. When mass is conserved, it means that the mass quantity of system 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 **u** are the density, time and velocity respectively and $\nabla \cdot (\rho \boldsymbol{u})$ represents the overall rate of mass additions per unit volume. [28]. 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

$$\nabla \cdot \boldsymbol{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{2.2}$$

where u, v and w are the velocity components in the x, y and z direction, respectively.

The conservation of momentum is nothing more than Newton’s second law, $F = ma$, applied to fluid elements. For an incompressible flow, the conservation of momentum can be described as

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

This equation is known as the Navier-Stokes equations [24], where μ is known as the dynamic viscosity, which is a material property. 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-equation 2.4 and the continuity equation 2.2 are the basis for describing the motion of an incompressible flow of a homogeneous fluid.

As mentioned earlier, turbulent flows involve a very wide range of active spatial and temporal scales. In combination with the chaotic behavior 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, this requires a lot of computational time, despite the growth of computer power in the last decades. Therefore, to deal with the complexity of the problem, several theoretical and computational approaches have been introduced to characterize turbulence. Among these ”simplified engineering approximations”, Reynolds-averaged Navier-Stokes (RANS) and large-eddy simulation (LES) approached are most common [9]. In the following sections, RANS and LES are discussed in more detail.

2.2. RANS

Since it is very expensive and time consuming to resolve all the scales of the Navier Stokes equations, simple models have been developed to approximate the turbulent motion. The first type of model we will discuss are based on the RANS equations and therefore are called RANS models. The RANS equations are obtained by splitting the instantaneous flow, u, into a mean quantity, $\overline{u}$, and its associated fluctuations, u'. This is called Reynolds decomposition, named after its originator Osborne Reynolds and can be written as

$$\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}'. \tag{2.5}$$

The time-averaged velocity can be simply obtained by

$$\boldsymbol{u}(\boldsymbol{x}) = \lim_{T \to \infty} \frac{1}{T} \int_t^{t+T} \boldsymbol{u}(\boldsymbol{x}, t) dt. \tag{2.6}$$

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

$$\frac{D\overline{u_i}}{Dt} = -\frac{1}{\rho}\frac{\overline{p}}{x_j} + \frac{\partial}{\partial x_j}(\nu\frac{\partial \overline{u_i}}{\partial x_j} - \overline{u_i' u_j'}), \tag{2.7}$$

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0. \tag{2.8}$$

Note that the RANS equations include a new term $\overline{u_i' u_j'}$, called the Reynolds stresses. These Reynolds stresses cause that we have six additional unknowns making it impossible to solve the equations, which is referred to as the turbulence closure problem. Hence, additional models for the Reynolds stresses are required.

2.2.1. Linear Eddy-Viscosity Models

A majority of the turbulence models, which models the Reynolds stresses in the RANS equations, are the linear eddy viscosity models. These models rely on a linear constitutive relationship, also known as the Boussinesq approximation, yielding

$$\overline{u_i' u_j'} = 2\nu_t S_{ij} - \frac{2}{3}k\delta_{ij}. \tag{2.9}$$

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

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right), \tag{2.10}$$

k is the mean turbulent kinetic energy

$$k = \frac{1}{2}(\overline{u_i' u_i'}), \tag{2.11}$$

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 have been derived. These eddy viscosity models can be classified in the amount of additional equations used. 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 and hence, their application is limited to very simple flow geometries. An example of such a model is the Prandtl mixing-length model for shear flows given by the equation:

$$\nu_t = \left|\frac{\partial \overline{u}}{\partial y}\right| l_m^2, \tag{2.12}$$

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.

In one-equation models, one is typically solving a transport equation for the turbulence kinetic energy k or eddy viscosity. A good example for such models is Prandtl's one-equation model that solves the eddy viscosity by

$$\nu_t = l_m\sqrt{k}, \tag{2.13}$$

where k is described by the PDE

$$\frac{\partial k}{\partial t} + \overline{u_j}\frac{\partial k}{\partial x_j} = R_{ij}\frac{\partial \overline{u_i}}{\partial x_j} - \epsilon + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{2.14}$$

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. The first developed two-equation transport model is a $k-\omega$ model, in which the transport equations of the eddy frequency ω and the turbulent kinetic energy are considered as

$$\frac{\partial k}{\partial t}+\overline{u_j}\frac{\partial k}{\partial x_i}=R_{ij}\frac{\partial \overline{u_i}}{\partial x_j}-\beta^* k\omega+\frac{\partial}{\partial x_j}\left[\left(\nu+\nu_t\sigma^*\right)\frac{\partial k}{\partial x_i}\right], \tag{2.15}$$

$$\frac{\partial \omega}{\partial t}+\overline{u_i}\frac{\partial \omega}{\partial x_i}=\gamma\frac{\omega}{k}R_{ij}\frac{\partial \overline{u_i}}{\partial x_j}-\beta\omega^2+\frac{\partial}{\partial x_i}\left[\left(\nu+\nu_t\sigma\right)\frac{\partial \omega}{\partial x_i}\right], \tag{2.16}$$

which contains the five closure coefficients β, β^*, γ, σ and σ^*. The closure relationship for this model is

$$\nu_t=\frac{k}{\omega}. \tag{2.17}$$

An other widespread two-equation model is the $k-\epsilon$ model, which solves a transport equation for both the turbulent kinetic energy and the dissipation, which are modeled as:

$$\frac{\partial k}{\partial t}+\overline{u_j}\frac{\partial k}{\partial x_j}=R_{ij}\frac{\partial \overline{u_i}}{\partial x_j}-\epsilon+\frac{\partial}{\partial x_j}\left[\left(\nu+\frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right], \tag{2.18}$$

$$\frac{\partial \epsilon}{\partial t}+\overline{u_j}\frac{\partial \epsilon}{\partial x_j}=C_1\frac{\epsilon}{k}R_{ij}\frac{\partial \overline{u_i}}{\partial x_j}-C_2\frac{\epsilon^2}{k}+\frac{\partial}{\partial x_j}\left[\left(\nu+\frac{\nu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right], \tag{2.19}$$

where C_1, C_2, σ_k and σ_ϵ are constant. The eddy viscosity is calculated as

$$\nu_t=C_\mu\frac{k^2}{\epsilon}. \tag{2.20}$$

2.3. LES

Although the focus in this project is on RANS models, for completeness a brief explanation of LES models is given as well. The larger eddies in turbulent flows obtain their kinetic energy from the bulk fluid energy, containing most of the turbulent kinetic energy and transfer this energy to the smaller eddies by stretching and breaking them up [28]. 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 transfer it back to the fluid. The smaller eddies are statistically isotropic and therefore, more universal and more independent of the boundary conditions and the mean flow velocity than the larger eddies. Bases on this theory, large eddy simulation 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 scales, which are modeled resulting in

$$\boldsymbol{u}=\overline{\boldsymbol{u}}+\boldsymbol{u}'. \tag{2.21}$$

The larger eddy velocity is obtained by employing a filtering operation

$$\overline{\boldsymbol{u}}(\boldsymbol{x},t)\equiv\int \boldsymbol{u}(\boldsymbol{x}',t)G(\boldsymbol{x}-\boldsymbol{x}')d\boldsymbol{x}', \tag{2.22}$$

where $G(\boldsymbol{x}-\boldsymbol{x}')$ is a filter function. The equation 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, an order or two of magnitude more time intensive than RANS.

2.4. Anisotropy Tensor

The Reynolds stresses, $\tau_{ij}=\overline{u_i'u_j'}$ can be divided into an isotropic and anisotropic part:

$$\tau_{ij}=\frac{2}{3}k\delta_{ij}+a_{ij}. \tag{2.23}$$

Here, a_{ij} is the Reynolds stress anisotropy tensor and $k = \frac{1}{2}\text{trace}(\tau_{ij})$ is the turbulent kinetic energy. The anisotropic part of the Reynolds stress is the most important part, since only this part is effective in transporting momentum. Equation 2.23 can be rewritten to define the non-dimensional anisotropic Reynolds stress tensor, b_{ij}, as

$$b_{ij} = \frac{\tau_{ij}}{2k} - \frac{1}{3}\delta_{ij}. \tag{2.24}$$

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

2.4.1. Realisability

From the properties of the Reynolds stresses τ_{ij} and its anisotropy b_{ij}, physical constraints can be derived [15]. Starting with the definition that a square matrix $\boldsymbol{A}$ 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

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

and is still positive semi-definite after taking the temporal average $\tau_{ij} = \overline{u_i' u_j'} = \frac{1}{n}\sum_n u_i' u_j'$, since all the time-samples adhere this property. Since the Reynolds stress are 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:

$$\tau_{\alpha\alpha} \geq 0 \quad \forall \alpha \in \{1,2,3\}, \qquad \det(\tau) \geq 0, \qquad \tau_{\alpha\beta}^2 \leq \tau_{\alpha\alpha}\tau_{\beta\beta} \quad \forall \alpha \neq \beta \tag{2.25}$$

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

$$\lambda_i = \frac{\phi_i}{2k} - \frac{1}{3}. \tag{2.26}$$

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 2.25, that the off-diagonal components of b_{ij} are in $[-\frac{1}{2}, \frac{1}{2}]$.

With anisotropy-invariant maps, the anisotropy can visualised. There exist two independent invariants of the anisotropy tensor, which can quantify the level of anisotropy of turbulent quantities:

$$II = b_{ij}b_{ji}, \qquad III = b_{ij}b_{in}b_{jn}. \tag{2.27}$$

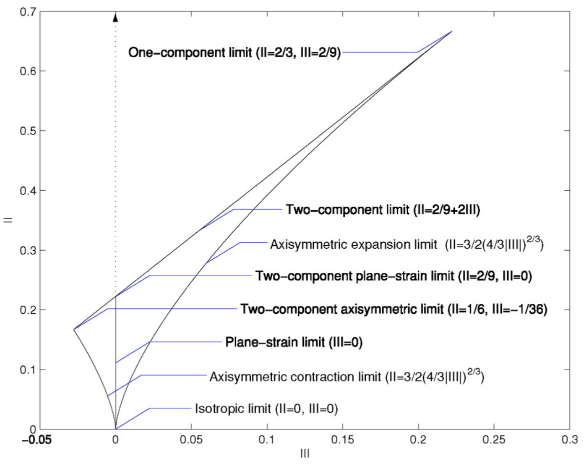
These invariants, representing the realizable states of turbulence anisotropy, can be plotted in the **II-III** plane, which was introduced by Lumley and Newman [19], [20]. The invariants fall within a triangular domain, corresponding to the constraints mentioned previously. Rewriting the domains of the of the anisotropy tensor, the following nonlinear relationships can be constructed:

$$II \geq \frac{3}{2}\left(\frac{4}{3}|III|\right)^{2/3} \qquad II \leq \frac{2}{9} + 2III \tag{2.28}$$

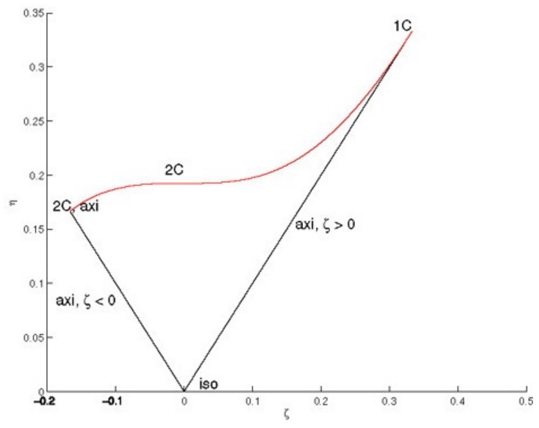
Lumley [7] also suggested another representation of the anisotropy which is denoted by $\eta - \zeta$:

$$\zeta^3 = III/2, \qquad \eta^2 = -II/2. \tag{2.29}$$

These constraints can maybe in someway be applied during the sparse symbolic regression, to obtain realisable models.



(a) Anisotropy-invariant map based on (III, II), taken from [1, p. 6]



(b) Anisotropy-invariant map based on (η, ζ), taken from [1, p. 2]

3 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 machine learning 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. Steven 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” [5]. 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 [3]. The flexible modeling framework of ML can be applied to many different challenges in fluid mechanics, including governing equation discovery, flow decomposition, producing reduced-order models and flow control. In this chapter, a rough overview of the state-of-the-art of ML in turbulence modeling is given and we explicitly focus on the augmentation of RANS models with ML.

3.1. Machine Learning

Machine Learning is an umbrella term for a wide range of techniques within artificial intelligence. ML can be generally defined as ”the field of study that gives computers the ability to learn without being explicitly programmed” [29]. The main task of ML is to develop learning algorithms that build models from data. The learning algorithm receives training data to obtain a model that can make prediction on new data [45].

ML is able to incorporate high-dimensional data and nonlinear modeling assumption. Therefore, ML-based methods are attractive tools that can help to extract more, hidden knowledge from data and thus are a great flexible modeling technique. ML can be applied to different RANS problems, dependent on what specific part of the RANS problem is modeled. A first approach for the use of ML is to replace the unclosed Reynolds stresses by an a priori determined function g with parameters θ and use ML to fit those parameters [3]. Alternatively, we can directly predict the closure term. So no priori determined function g is assumed, but is obtained by the ML method. A third approach is to model the full governing PDEs and thus directly approximate the solution $\overline{\boldsymbol{u}}$. In the next sections, selected ML algorithms are introduced and some successful application of that ML method to turbulence modeling is 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. However, no converged state of the art has been reached yet.

ML algorithms can be categorized into supervised learning, unsupervised learning and semi-supervised learning. In supervised learning, the objective is to construct a function, which maps the inputs to given outputs. Common supervised machine learning methods are linear regression, random forests, SVMs and artificial neural networks. These methods are also used in turbulence modeling, and will be explained in more details in upcoming sections.

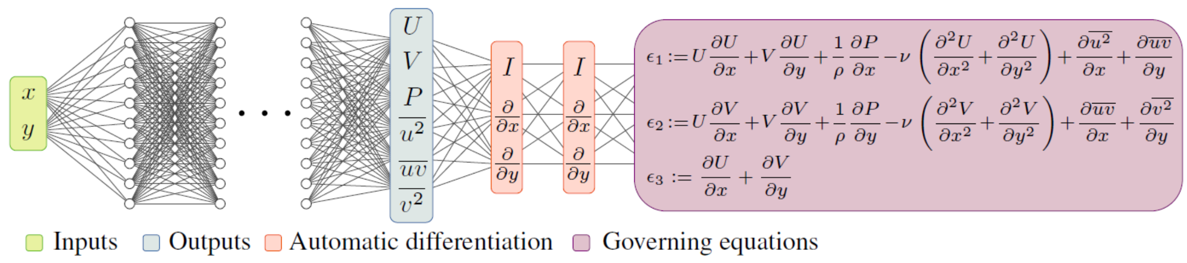

x
y
U V P $\overline{u^2}$ $\overline{uv}$ $\overline{v^2}$
I $\frac{\partial}{\partial x}$ $\frac{\partial}{\partial y}$
I $\frac{\partial}{\partial x}$ $\frac{\partial}{\partial y}$
$\epsilon_1 := U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial x} - \nu\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) + \frac{\partial \overline{u^2}}{\partial x} + \frac{\partial \overline{uv}}{\partial y}$
$\epsilon_2 := U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial y} - \nu\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) + \frac{\partial \overline{uv}}{\partial x} + \frac{\partial \overline{v^2}}{\partial y}$
$\epsilon_3 := \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y}$
Inputs
Outputs
Automatic differentiation
Governing equations

Figure 3.1: PINN used for solving the RANS equations for a general two-dimensional set-up, taken from [11, p. 2]

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 often is used to find unknown correlations and patterns in datasets. K-means clustering is an example of such a technique, which minimizes the average variance of k clusters.

3.1.1. Neural Networks

Artificial neural networks consist of simple elements which are parallel interconnected in a hierarchical, layered organization [16]. Those simple elements are called neurons, and are intended to work similarly as neurons in the biological nervous systems. Artificial neural networks are characterised by their flexibility and hence many variations of neural networks exist. The classic 'feed forward' neural network 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 neural network can be seen in Figure **??** (a). 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. Commonly used activation functions are the ReLU function, $g(x) = max\{x, 0\}$, the sigmoid function, $g(x) = 1/(1 + \exp^{-x})$, or the hyperbolic tangent function, $g(x) = \tanh(x)$. 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 layers. 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.

Neural networks are generally one of the most widespread ML algorithm. Hence also in modeling turbulence it is a widely researched method. Tracey *et al.* were one of the pioneers in modeling turbulence with ANNs [33]. They used Neural Networks to reconstruct the closure term of the RANS equation. More specifically, they discover formulation of the terms that have been explicitly modeled in the Spalart-Allmaras turbulence model. More researchers followed in the use of neural networks for modeling closure terms in RANS or LES models [12], [22], [34], [41] and [2]. A study of particular interests is the one of Ling *et al.* [18]. They used deep neural networks to learn a model for the Reynolds stress anisotropy tensor, but embed physical knowledge by preserving Galilean invariance of the neural network predictions.

Another, currently new and hot approach, are Physics Informed Neural Networks (PINN's) [26], which Eivazi *et al.* introduced in turbulence modeling [11]. This black box algorithm completely by-passes the closure problem by directly solving the RANS equations with a neural network. As can be seen in Figure 3.1, spacial coordinates are taken as input and the full flow field variables are the output of the neural network, taking the residual of the Navier Stokes equations as the loss function.

Despite the flexibility of neural networks and their ability to represent complex interaction of different variables, several disadvantage impede the application in engineering problems. These include the lack

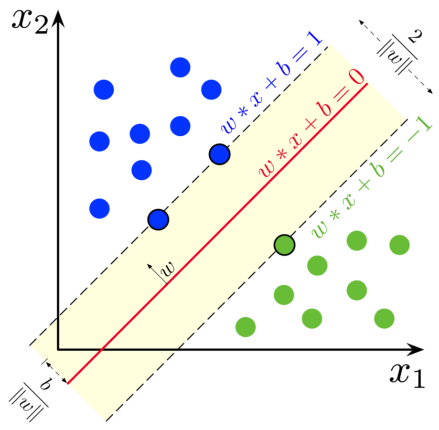


Figure 3.2: Visualisation of linear SVM

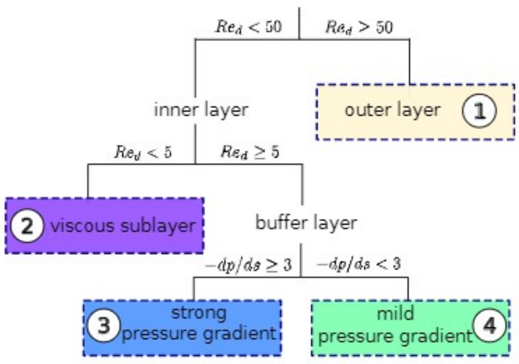


Figure 3.3: Example of an decision tree, taken from [36, p. 16]

of generelizability [32], and the often large required data-set imposing still high computation times.

3.1.2. Support Vector Machine

The basic idea of a support vector machine (SVM) algorithm is to find a hyperplane that separates the data into two distinct classes. This hyperplane is given by

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0, \tag{3.2}$$

where x is the vector with the different input features. With this hyperplane, new unseen data can be classified based on which side of the hyperplane the respective data point is located. The separating hyperplane is optimal if the distance between the data points of both sets to the hyperplane is maximised, as shown by figure 3.2. Obviously, many datasets are not linearly separable. In that case the dataset can be transformed by a nonlinear mapping of the input space to a high-dimensional feature space, in which the data will be linearly separable.

SVMs are mainly used for supervised classification tasks. Ling *et al.* used an SVM to classify regions in a flow where RANS will have a higher uncertainty because of specific modeling assumptions [17]. For the regions indicated with a higher uncertainty, the error could be mitigated by applying relevant corrections or models.

3.1.3. Decision Trees

A decision tree is a very intuitive ML method, since it makes decision based on a tree structure. It typically consists of one root node, multiple internal nodes and multiple leaf nodes. Each node corresponds to a decision rule, leading to an additional decision rule or the final decision outcomes, the leaf nodes, see Figure 3.3 for an example. The ML algorithm aims to produce a tree with decision rules that divides the training data most efficiently. 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 generalizability [14]. Wang *et al.* used such random forests to reconstruct discrepancies in RANS modeled Reynolds stresses [36]. Random forests are able to regress high dimensionality of the feature space and provides physical insights of the predictions and the importance of features as well as quantified uncertainties. However, Wang *et al.* did not implement their model in an RANS solver, hence the improvement of the propagated velocities from the corrected Reynolds stress field could not be guaranteed.

3.1.4. Gene Expression Programming

Gene Expression Programming (GEP) is an Evolutionary Algorithm (EA), that mimic nature's survival of the fittest, to end up with an algebraic equation that reproduce the data the best. GEP is encoded in

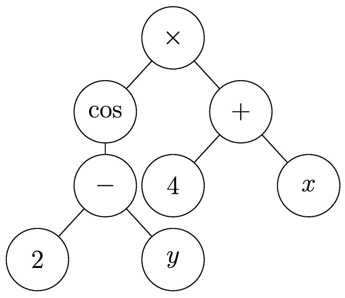


Figure 3.4: Example of Gene Expression Programming

simple linear structures, the chromosomes, containing one or more genes, which represent mathematical operators and constants. In the training process, these chromosomes are randomly mutated and the best ones are selected. The algorithm returns a mathematical equation, which is tangible and can be implemented into a CFD code. Weatheritt and Sandberg used this algorithm to model the anisotropy of the turbulent stress tensor [37]. Their results showed that the framework is a viable methodology for RANS closure development and provided inspiration and handles for other researches, like Zhao *et al.* [43] and Reissmann *et al.* [27]. Zhao applied GEP to LES model development and Reissmann extended the GEP method of Weatheritt and Sandberg but integrated, trained and evaluated the models directly in RANS solvers.

3.1.5. Sparse Symbolic Regression

Symbolic regression is a type of regression analysis that tries to find a mathematical expression that fits a given dataset best. GEP is a form of symbolic regression and 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 discovers for each run another model with different mathematical expressions because of its non-deterministic behavior. A deterministic variant is sparse symbolic regression based upon the data-driven technique presented by Brunton *et al.* [6]. They use data to discover governing equations of nonlinear, dynamical systems and considers dynamical systems of the form

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t)). \tag{3.3}$$

they 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, consisting of nonlinear candidate functions dependent on state $\boldsymbol{x}$. This library may consist of constant, polynomial, trigonometric or any other terms:

$$\Theta(\boldsymbol{x}) = \begin{bmatrix} 1 & \boldsymbol{x} & \boldsymbol{x}^2 & \dots & \sin \boldsymbol{x} & \cos \boldsymbol{x} \end{bmatrix}. \tag{3.4}$$

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

$$\dot{x} = \Theta(x)\Xi. \tag{3.5}$$

This way of finding governing equations has inspired Schmelzer *et al.* [30] and Beetham *et al.* [4], who both adopted the technique to find a mathematical expression for the anisotropic Reynolds stress.

3.2. Challenges and Opportunities

ML found its way into turbulence modeling just recently, but major progress already has been made. However, there are still challenges left as Andrea Beck *et al.* noted the following [3]:

- The need for data.
- Consistent data and model.
- Inclusive optimization
- Algorithmic and hardware considerations.

- Physical constraints
- Generlizability, interpretability, and convergence
- Efficiency and ease to use

From these challenges, we want the focus on the last three points. These challenges were taken into account when we selected the machine learning technique.

Beck also summed up the following opportunities for ML-augmented turbulence modeling:

- A new paradigm
- feature extraction
- Flexibility
- Incorporating discretization effects
- Exploiting existing turbulence data
- Arbitrary input features

From these opportunities we hope to take advantage of ML's flexibility and its opportunity to extract features.

3.3. Method Selection

Taking into account the end goal of this project, developing an interpretable data-driven RANS turbulence model, which contains physical knowledge and is generalizable and robust, in combination with the challenges and opportunities of ML, sparse symbolic regression was chosen as machine learning method to obtain a data-driven RANS turbulence model. In the rest of this section I want to elaborate the arguments for selecting this approach over other Machine Learning techniques.

To start with the argument 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. Physical knowledge can be extracted from the algebraic equation, which can give insight into turbulence. This is in contrast with models based upon NNs, which acts as a ”black box” and cannot be expressed in a compact algebraic form [4]. For this reason, other approaches as gene expression programming and random forest regression increased in popularity. Gene expression programming also produces an algebraic equation. Random forest regression however does not have an algebraic equation as result but it has an intuitive way of selecting features and creating models.

Secondly, sparse symbolic regression provides the possibility to inform physical knowledge. For instance, Galilean invariance can be ensured by carefully constructing the library of candidate function and structuring of the optimization cost functional.

At last, 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 NNs. The simple algebraic equation obtained by sparse symbolic regression is also easier and more efficient to integrate into existing CFD solvers. NNs and RFs on the other hand are much more complicated to integrate into a solver and therefore, the velocity cannot be computed. Wang *et al.* faced this problem in their study on turbulence modeling with a random forest method [36].

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

4 Methodology

In this chapter the methodologies used and developed are discusses. We start in section 4.1 by providing the overall frame-work of the modeling procedure. Thereafter a detailed description of the employed input features is given in section 4.2, followed with explanation of sparse symbolic regression. At last we expand the sparse symbolic regression method by adding physical constraints.

4.1. Modeling Framework

We aim to develop an improved algebraic closure model for the Reynolds stresses that appear in the RANS equations (2.7). For this, we start by decomposing the Reynolds stress tensor into its isotropic part, $\frac{2}{3}k\delta_{ij}$, and anisotropic part, $a_{ij} = 2kb_{ij}$, yielding:

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

As outlined previously, a linear eddy viscosity model, based on the Boussinesq equation is frequently used to model this anisotropic part as $b_{ij} = \frac{\nu_t}{k}S_{ij}$.

Pope recognized that a more general nonlinear representation of the anisotropic stress tensor is a finite tensor polynomial, dependent on both the normalised strain and the rotation rate tensor, S_{ij} and R_{ij} respectively [25]. 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, that describes the anisotropic stress tensor as

$$b_{ij}(S_{ij}R_{ij}) = \sum_{n=1}^{10} G^{(n)}(I_1, ..., I_5)T_{ij}^{(n)}. \tag{4.2}$$

Here, the coefficients $G^{(n)}$ may be functions of the five invariants $I_1, ..., I_5$ listed in table 4.2. The ten tensor bases that are used in this equation are listed in table 4.1. For these tensors, the normalised strain and rotation tensors, S_{ij} and R_{ij}, are used. Using equation 4.2 has as advantage that Galilean invariance can be ensured, which means that when the coordinate frame is rotated the anisotropy tensor is also rotated by the same angles. 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.

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

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

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

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^*, and take the nonlinear part as subject of modeling. The linear part of the anisotropic stress can be taken as a standard LEVM, such that

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

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 [39]. 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 stabilize the RANS simulation and prevent the error amplification of the velocities.

In this research the nonlinear part of the anisotropic stress tenor is the subject of modeling. This tensor is calculated with equation 4.3 in which the linear part b^o_{ij} is derived from the output of the $k-\omega$ 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 2.16. For the complete anisotropic stress tensor b_{ij}, DNS data is used and hence our target of modeling is derived by

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

in which *DNS* and *RANS* indicates that data from a DNS database or a RANS simulation is used. This way of calculating the nonlinear part of the anisotropic stress, is inspired by the research of Beetham *et al.* [4] and Wu *et al.* [38]. A slightly different approach is done by Schmelzer *et al.* [30] and Weatheritt *et al.* [37]. Instead of calculating the linear part with RANS data, k, ν_t and S_{ij} are derived from DNS data. However, this derivation is more complicated to implement and hence will not be used in the research.

With sparse symbolic regression we aim to find an algebraic expression for the nonlinear anisotropic stress tensor b^* using equation 4.2. The model discovery consists of different steps. First, a library of input features needs to be created based on data from a RANS simulation. Thereafter, sparse symbolic regression can be conducted which again consists of two steps. First, the functions are selected with a sparse regression function. Secondly, the coefficients of the selected functions are calibrated with a ridge regression functions. At last the mean velocities have to be derived by solving the 'new' RANS equations with the discovered model for the anisotropic stress, which is referred to as the 'propagation' of the new Reynolds stress field to mean velocities. A complete modeling overview can be seen in Figure 4.1

The next sections, will explain which input features we will use and how sparse symbolic regression will find an algebraic expression for the anisotropic stress tensor.

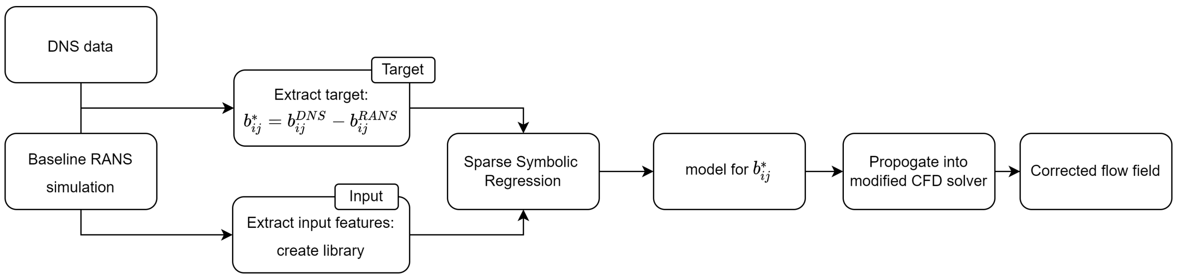

DNS data
Target
Extract target:
$b^*_{ij} = b^{DNS}_{ij} - b^{RANS}_{ij}$
Baseline RANS simulation
Sparse Symbolic Regression
model for b^*_{ij}
Propogate into modified CFD solver
Corrected flow field
Input
Extract input features: create library

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

4.2. Input Features

Using the right input features can have significant influence on the performance of the learned model. Already quite some effort has been made in choosing proper mean flow features in previous research. Yin *et al.* noted that there are three basic principles during the construction of input features [42].

1 Completeness: the set of input features should include all possible information that is relevant for modeling the Reynolds stresses
2 Compactness: invalid or redundant information should be excluded as much as possible
3 Realizability: selected features should be consistently effective under various circumstances, such as different flow directions.

We start by considering feature selection based on tensor analysis. Since we rely on the nonlinear eddy viscosity concept of Pope and aim to find the coefficients in equation 4.2, we start by including the invariants of table 4.2 as input features. However, it is possible to include additional features to predict the coefficients of the basis tensor. Instead of $G^{(n)}(I_1, ..., I_5)$, the tensor basis coefficients can be a function of m different input features, $G^{(n)}(q_1, ..., q_m)$. Wang *et al.* [36] expanded the dependency of the Reynolds stress by adding pressure gradients and turbulent kinetic energy gradients and transforming those into corresponding anti-symmetric tenors A_p and A_k:

$$\begin{aligned} A_p &= -I \times \nabla p \\ A_k &= -I \times \nabla k \end{aligned} \tag{4.5}$$

The tensors A_p and A_k can form an integrity basis with 41 components. To use these tensors as input features they took the first invariant, the trace of each tensor.
However, in our opinion these invariants are exotic and hard to interpreted. Furthermore, the cross product is rather complex to implement in OpenFOAM. Therefore we decided to not use these invariants in the rest of this project, but only use the invariants I_1 up till I_5 from table 4.2.

Besides the set of invariants, extra scalar features which are more physically interpretable can be used, which were obtained as well from Wu *et al.*. All the features are normalized to fall within the range $[-1, 1]$ as done by Ling and Templeton [17] according the formula:

$$q_\beta = \frac{q_\beta}{|q_\beta| + |q_\beta^*|}. \tag{4.6}$$

Here q_β are the raw values of the features and q_β^* are the corresponding normalisation factors. All the physical features and their corresponding normalization factors are stated in table 4.3, including a short description of the features.

Table 4.3: Features used as input

index	Features	Normalization	Comment
q_1	$\frac{1}{2}(\lvert\lvert R\rvert\rvert^2 - \lvert\lvert S\rvert\rvert^2)$	$\lvert\lvert S\rvert\rvert^2$	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	$\frac{1}{\lvert\lvert S\rvert\rvert}$	ratio of the turbulent time scale to the mean flow time scale
q_5	$\overline{u}_i \frac{\partial k}{\partial x_i}$	$\lvert\overline{u'_j u'_k} S_{jk}\rvert$	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	$\lvert\overline{u}_i\overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j}\rvert$	$\sqrt{\overline{u}_i\overline{u}_i\overline{u}_i \frac{\partial \overline{u}_i}{\partial x_j}\overline{u}_k \frac{\partial \overline{u}_k}{\partial x_j}}$	non-orthogonality between velocity and its gradient

4.3. Sparse Symbolic Regression

Sparse symbolic regression requires a library of candidate function. From this library, the relevant candidate functions explaining the data, are selected by using a sparse promoting regression technique. Hence, constructing the library is essential and can be done through the following steps

1. We start by combining the pre-defined features with pre-defined exponents or trigonometric terms, resulting in library $\mathcal{B}_1$.
2. Thereafter we can expand the library, by making the features interactive. This is done by multiplying the features from $\mathcal{B}_1$ with other features from $\mathcal{B}_1$, resulting in $\mathcal{B}_2$ and once again multiply the features from $\mathcal{B}_1$ with the features in $\mathcal{B}_2$, giving $\mathcal{B}_3$.
3. The library of features is given by $\mathcal{B} = \mathcal{B}_1 + \mathcal{B}_2 + \mathcal{B}_3$.
4. The library of candidate functions $\mathcal{C}$ is obtained by multiplying $\mathcal{B}$ with each base tensor $T_{ij}^{(n)}$ from table 4.1.

Given the constructed library $\mathcal{C}$, we aim to form a linear model to regress the target data, the nonlinear part of the anisotropic stress b^* by finding the coefficient vector Θ, such that

$$b^* = \mathcal{C}\Theta.$$

Following the methodology of Schmelzer *et al.*, the model-discovery procedure to find the coefficient vector Θ, consists of two parts [30]. First, a model selection step is performed, followed with a model inference step. In the model selection step, the goal is to end up with a smaller subset of candidate functions, such that a simple model can be formed. For this, sparsity-promoting regularisation of a least-squares optimisation problem is used. A common used regression method that incorporates this regularisation is LASSO regression. The objective of LASSO is to solve the minimization problem

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

where the penalty term $\lambda||\hat{\Theta}||_1$ represents the l_1-norm of the coefficient vector, weighted with regularisation parameter λ. This l_1-norm allows only a few nonzero coefficients while shrinking the rest to zero. Larger values for λ results in more zero valued coefficients in Θ, while smaller values for λ increases the amount of nonzero coefficients. Note that when λ is set to zero, the function 4.7 becomes the least-squares objective function.

Another sparsity-promoting regularisation of the least-squares optimisation problem is the elastic net. The elastic net is an extension of the Lasso by adding an addition l_2-norm regularisation term resulting in the minimization problem

$$\Theta = \arg\min_{\hat{\Theta}} ||\mathcal{C}\hat{\Theta} - b||_2^2 + \lambda\rho||\hat{\Theta}||_1 + 0.5\lambda(1-\rho)||\hat{\Theta}||_2^2. \tag{4.8}$$

Here, the l_2-norm, known as 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 (4.8), $\rho \in [0,1]$ is a mixing parameter between the l_1 and l_2-norm.

Given the Lasso regression method, 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 $\Theta^{(i)}$ as a solution of equations 4.7 with λ_i is found. However, different values of λ_i can produce a coefficient vector $\Theta^{(i)}$ with the same entries equal to zero. Therefore, the final step of the model selection is to filter out the unique models with different nonzero entries of coefficient vector Θ. We aim to try the two different regularisation functions, LASSO and elasticnet, and compare the selected models.

After selecting a set of unique abstract models, an additional regression is performed for each model and its selected candidate functions, which is called the model inference step. The main reason for this step, is that in the model selection step, the candidate functions are standardised. This is done since the relevance of each candidate should not depend on is magnitude. To create a model with the correct units, the additional regression is done using the unstandardised candidate functions. The additional regression is done by using the Ridge regression

$$\Theta = \arg\min_{\hat{\Theta}} ||\mathcal{C}\hat{\Theta} - b^*||_2^2 + \lambda_r||\hat{\Theta}||_2^2. \tag{4.9}$$

Here, the elements of Θ associated with the inactive candidates, as found in previous step, are zero and are not modified during this regression step. The l_2-norm regularisation term $\lambda_r||\hat{\Theta}||_2^2$, causes the magnitude of the nonzero coefficients to shrink. In general, high values for the weighting parameter λ_r lead to a lower magnitude of the coefficients. This is 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. A value of 0.1 for λ_r seemed to produce reasonable models.

In the end we have created a coefficient vector Θ to retrieve the symbolic expression for the anisotropic stress tensor b^*.

4.4. 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 into a CFD solver, which 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 in the velocity field. In this project the open source software OpenFOAM is used. The integration of the learned model in OpenFOAM should also expose the model shortcomings, such as sensitivity or stability issues.

To obtain a solver, which is able to compute the mean velocities using the correction of the Reynolds stress anisotropy tensor by the obtained model, the turbulence models in OpenFoam and the simpleFoam solver have to be modified.

The simpleFoam solver, solves the steady-state incompressible Reynolds Averaged Navier-Stokes equation. This equation, without any body forces, can be written as

$$U_j \frac{\partial U_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[-P + \nu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - R_{ij}\right]. \tag{4.10}$$

In the simple algorithm, this equation is iteratively solved by separating the part containing the pressure gradient term from the part containing the velocity. The pressure equation is solved in a separate file called pEqn.H, while the file UEqn.H contains the remaining terms of the momentum equation, i.e. $U_j \frac{\partial U_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - R_{ij}\right]$. This equation is denoted in the UEqn file as

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

Here, the term `turbulence->divDevReff(U)` references to another file, which depends on the used turbulence model. In our case, a linear eddy viscosity model is used, which dependents on the Boussinesq hypothesis. Since the Boussinesq assumption models the Reynolds stress, R_{ij}, as

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

the function `turbulence->divDevReff(U)` calculates:

$$\frac{\partial}{\partial x_j}\left((\nu + \nu_t)\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right). \tag{4.11}$$

Note that the term $\frac{2}{3}k\delta_{ij}$, is not there, 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 UEqn.H file the `turbulence->divDevReff(U)` will be removed and replaced by the direct calculation of the Boussinesq hypothesis and b^{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 $(\nu + \nu_t)$. The terms `nuEff()` and `k()` are called from another file dependent on the chosen turbulence model. Since the derivations of k and ν_t depend on the Reynolds stress as well, the turbulence models in which they are calculated have to be adjusted as well. When using the $k - \omega$ equations, the production term in those equation has to be modified to:

$$\mathcal{P} = -(2kb_{ij} + 2kb_{ij}^{\Delta} + \frac{2}{3}k\delta_{ij})\frac{\partial U_i}{\partial x_j}. \tag{4.12}$$

The k-omega equations in the turbulence model then looks like

```
tmp<volTensorField> gradU = fvc::grad(U);
bij_ = - nut/k_ * symm(fvc::grad(U));
#include "nonLinearModel.H"

volSymmTensorField R = (2.0/3.0)*k_*I - 2.0*nut*symm(fvc::grad(U)) + 2.0*k_*bDelta_;
volScalarField G(this->GName(),  -(R && fvc::grad(U)));

// Update omega and G at the wall
omega_.boundaryFieldRef().updateCoeffs();

// Turbulence specific dissipation rate 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_)
);

omegaEqn.ref().relax();
fvOptions.constrain(omegaEqn.ref());
omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef());
solve(omegaEqn);
fvOptions.correct(omega_);
bound(omega_, this->omegaMin_);

// 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_)
);

kEqn.ref().relax();
fvOptions.constrain(kEqn.ref());
solve(kEqn);
fvOptions.correct(k_);
bound(k_, this->kMin_);
```

5 Results

This chapter contains some first results of sparse symbolic regression for different cases, a channel flow and a flow through a channel with periodic hills.

5.1. Channel flow

As a starting case, a simple fully developed turbulent channel flow is considered. For this case the data of the DNS of a fully developed plane turbulent channel flow with a Re_τ of 392.24, conducted by Moser, Kim and Mansour is used as 'true' reference data [23]. The same flow case is simulated with an $k-\omega$ RANS model in OpenFoam. To obtain a flow with the same Re_τ, we used the physical parameters stated in table 5.1. Table 5.2 contains the numerical boundary conditions as used in the RANS simulation.

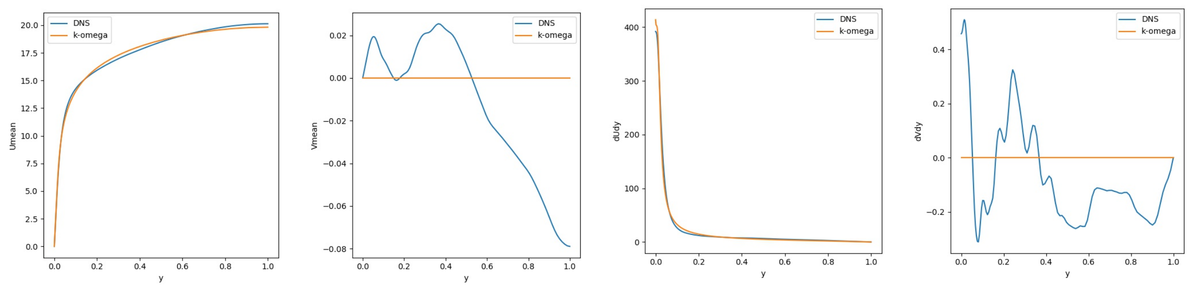
Table 5.1: Physical modelling

U_τ	$(1.0, 0.0, 0.0)$ $[m \cdot s^{-1}]$
ν	0.002532 $[m^2 \cdot s^{-1}]$
U_b	$(17.55, 0.0, 0.0)$ $[m \cdot s^{-1}]$

Table 5.2: Boundary and Initial conditions

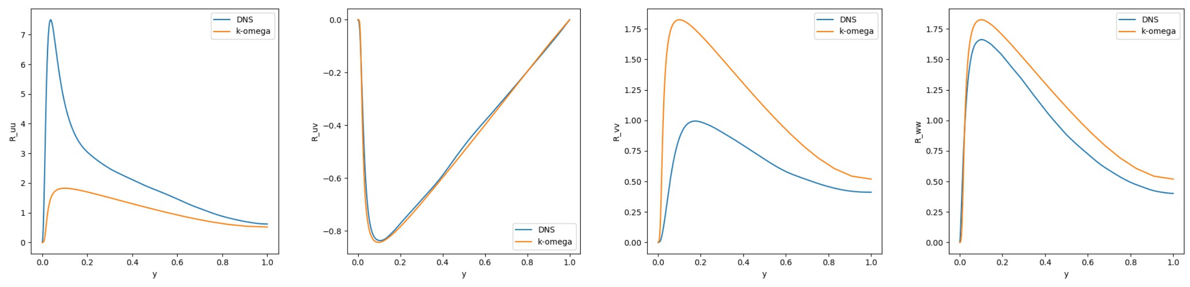
	Inlet	Outlet	Walls	Sides	Internal field
U	Cyclic	Cyclic	No Slip	empty	17.55
p	Cyclic	Cyclic	zeroGradient	empty	0
k	Cyclic	Cyclic	kqRWallFunction	empty	6.68e-05
ω	Cyclic	Cyclic	omegaWallFunction	empty	1.49e-02
ν_τ	Cyclic	Cyclic	nutUWallFunction	empty	4.48e-3

The velocity profiles for a cross-section of the channel of the DNS data and the RANS simulation are shown in figure 5.1. The Reynold stress profiles and the anisotropic parts of it are shown in figure 5.2 and 5.3. Here, the Reynolds stress of the $k-\omega$ model and its anisotropic part are obtained with the expressions, $R_{ij} = 2\nu_\tau S_{ij} + \frac{2}{3}k$ and $b_{ij} = -\frac{\nu_\tau}{k}S_{ij}$ respectively. The anisotropic stress part for the DNS case is calculated with $b_{ij} = \frac{R_{ij}}{2k} - \frac{1}{3}\delta_{ij}$.



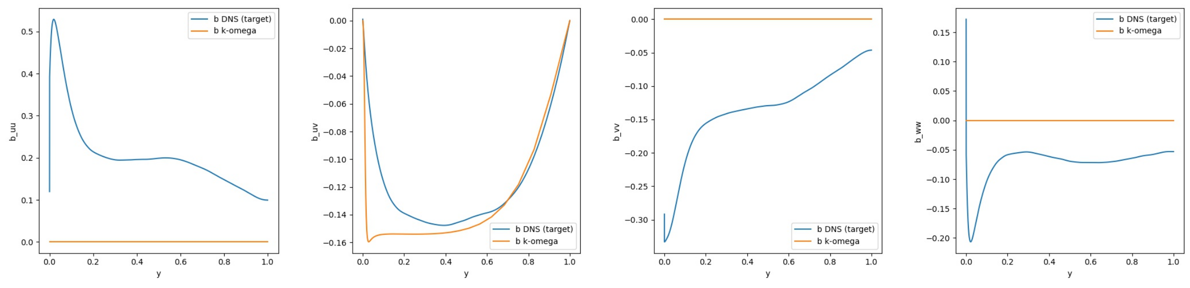
(a) U **(b)** mean V **(c)** $\frac{\partial U}{\partial y}$ **(d)** $\frac{\partial V}{\partial y}$

Figure 5.1: Velocity profiles of the DNS and RANS simulation of a turbulent channel flow with $Re_\tau = 395$.



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

Figure 5.2: Reynolds stress profiles of the DNS and RANS simulation of a turbulent channel flow with $Re_\tau = 395$.



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

Figure 5.3: Anisotropic part of the Reynold stress profiles of the DNS and RANS simulations of turbulent channel flow with $Re_\tau = 395$.

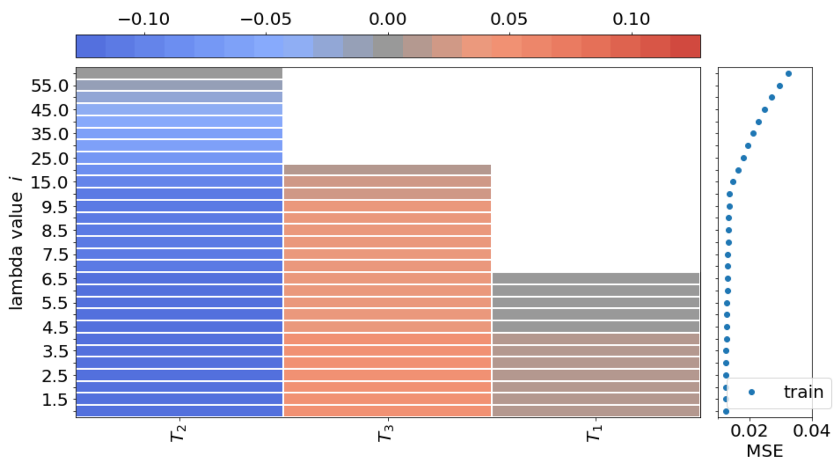
Table 5.3: Reduced set of invariant tensors for statistically two-dimensional flows

$T_{ij}^{(1)}$	S_{ij}
$T_{ij}^{(2)}$	$S_{ik}R_{kj} - R_{ij}S_{kj}$
$T_{ij}^{(3)}$	$S_{ik}S_{kj} - \frac{1}{3}S_{lk}S_{kl}\delta_{ij}$

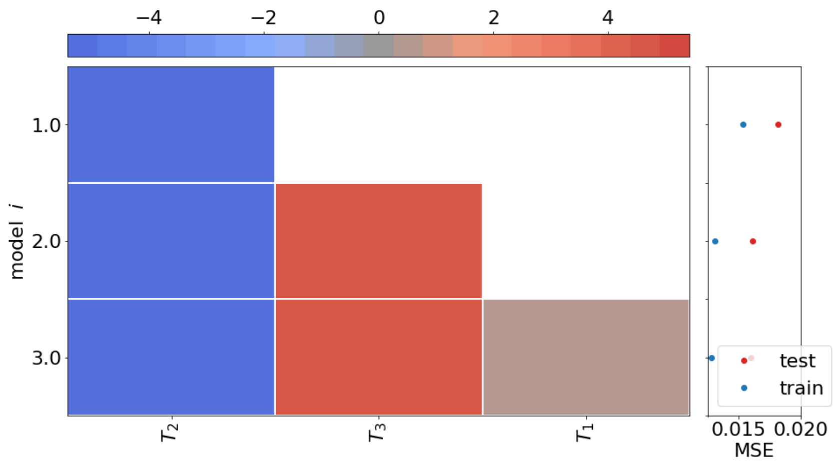
Sparse regression is applied to obtain an expression for the nonlinear anisotropic part of the Reynolds stresses in the RANS simulation, such that it simulates the nonlinear anisotropic part of the Reynolds stress of the DNS data. Since the channel flow is a two dimensional problem, there is no velocity component in the z-direction and the set of ten invariant tensors from table 4.1 reduces to three tensors. Hence, as an initial start, a library is created containing only the three invariant stress tensors shown in table 5.3 and assembled to matrix $\mathcal{C}$. So the library reads:

$$\mathcal{C} = \left[T_{ij}^{(1)}, T_{ij}^{(2)}, T_{ij}^{(3)}\right] \tag{5.1}$$

As described in section 4.3 a Lasso regression function with values of λ varying between 1 and 100 is solved to select the candidate functions. The different models for each λ function are shown in figure 5.4a. To solve the Lasso regression function, for this case the python package 'cvxpy' is used [8]. From this figure we see three models with different nonzero coefficients. These three models are 'calibrated' in the model inference step, in which an additional ridge regression is performed. The results of the model inference step for which we used a λ_r of 0.1 are shown in figure 5.4b.



(a) Model selection



(b) Model inference

Figure 5.4: Sparse Symbolic Regression for the turbulent channel flow

From the three found models, the one with the lowest test error has as symbolic equation

$$b^*_{ij} = 0.316T^{(1)}_{ij} - 5.421T^{(2)}_{ij} + 4.5167T^{(3)}_{ij}. \tag{5.2}$$

The anisotropic stress of the RANS simulation are corrected with the obtained equation (5.2) for the nonlinear part of the anisotropic stress tensor by

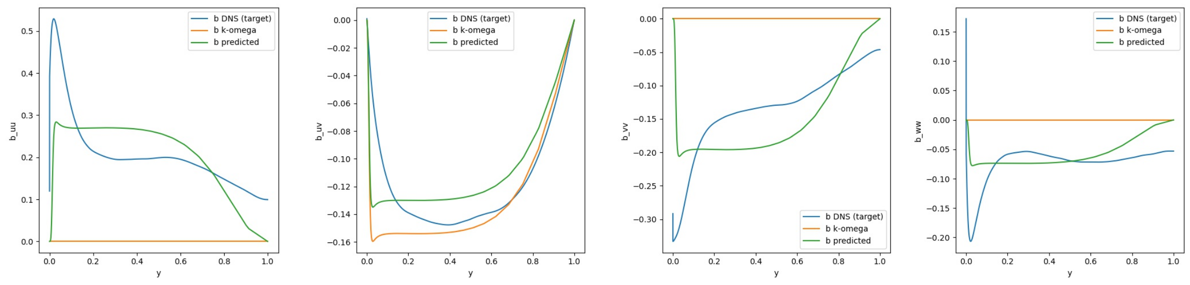
$$b_{ij} = -\frac{\nu_t}{k} S_{ij} + b^*_{ij}$$

and can be seen in figure 5.5, in which again the anisotropic stresses of the DNS and baseline RANS are shown. The Reynolds stresses of the RANS simulation are corrected with the obtained equation for the nonlinear anisotropic stresses by:

$$R^*_{ij} = 2k(b_{ij} + b^*_{ij} + \frac{1}{3}\delta_{ij}) \tag{5.3}$$

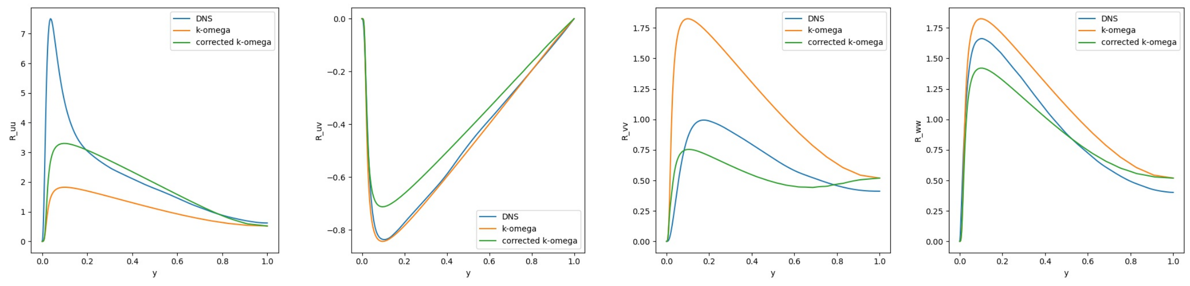
The corrected Reynolds stresses can be seen in figure 5.6, together with the Reynolds stresses of the DNS and baseline RANS simulation.

In figure 5.5 we can see that the all the corrected anisotropic stress profiles b_{ij} are almost identical to the anisotropic stress tensor b_{uv} of the RANS simulation, but than multiplied with some shrinking factor. This makes sense, since as candidate functions only the invariant tensors are used and hence only constants are found as coefficients. To improve the results, the library of candidate functions has to be expanded, so functions are found for the coefficients $G^{(n)}$ instead of constants. These functions could depend on the invariant I_1 and I_2, but also on other features as described in section 4.2.



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

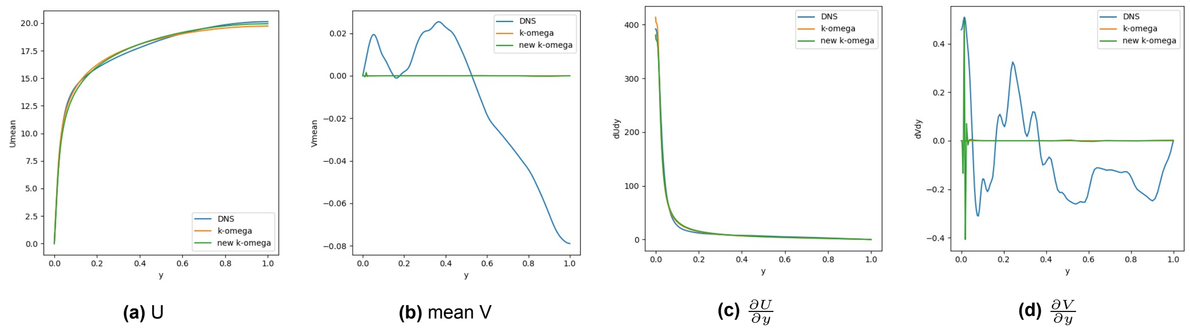
Figure 5.5: Anisotropic part of Reynolds stress profiles of the DNS and RANS simulation and the found expression of a turbulent channel flow with $Re_\tau = 395$.



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

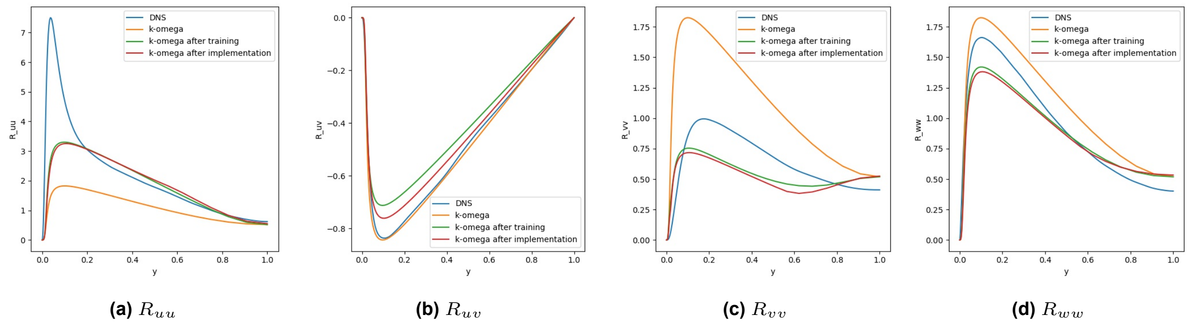
Figure 5.6: Reynolds stress profiles of the DNS and RANS simulation and the corrected Reynolds stress profile of a turbulent channel flow with $Re_\tau = 395$.

To see how the found equation for the anisotropic stresses propagates into the velocity profiles, we implement the anisotropic stress equations OpenFOAM as described in section 4.4. The velocity profile of this implementation are shown in figure 5.7. Figure 5.9 shows the corrected anisotropic stress profiles after the model of equations 5.2 is implemented in OpenFoam. For comparison, also the anisotropic stress profiles of the DNS and RANS simulations are plotted as well as the stress profiles obtained during the training. Figure 5.8 shows the corrected Reynolds stress profiles after the implementation in OpenFoam. Again, the Reynolds stress profiles of the DNS and RANS simulations are plotted as well as the stress profiles obtained during the training. Now the Reynolds stress of the implemented model is calculated as in equation 5.3. We see that the stresses from the propagated model are similar to the stresses obtained after training. This is ofcourse what we expect and suggest that the new model is correctly implemented. However, we do not see much change in velocity profiles, but the mean velocity profiles of the DNS and RANS simulation are already very similar. Hence, there is not much to improve for the velocity profiles.



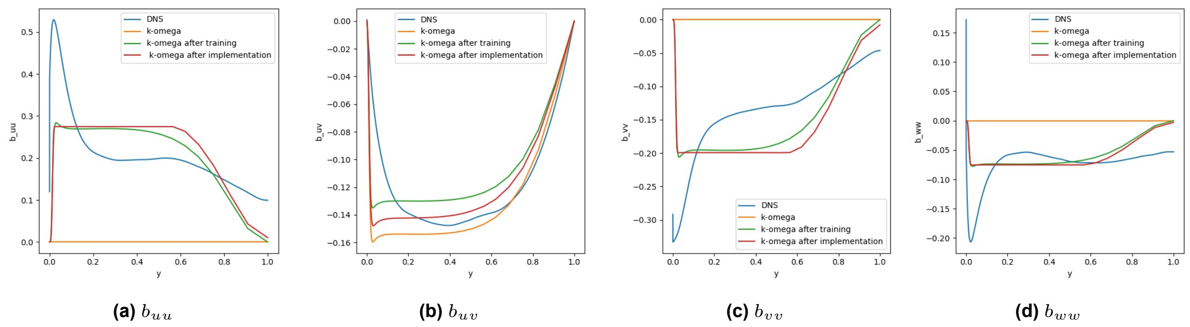
(a) U **(b)** mean V **(c)** $\frac{\partial U}{\partial y}$ **(d)** $\frac{\partial V}{\partial y}$

Figure 5.7: Velocity profiles of the DNS, RANS and corrected RANS simulation of a turbulent channel flow with $Re_\tau = 395$.



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

Figure 5.8: Reynolds stress profiles of the DNS, RANS simulation, the corrected Reynolds stress profile after training and the propagated stress profiles of a turbulent channel flow with $Re_\tau = 395$.



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

Figure 5.9: Anisotropic part of Reynolds stress profiles of the DNS, RANS simulation, the found expression of a turbulent after training, and the propagated anisotropic stresses of a channel flow with $Re_\tau = 395$.

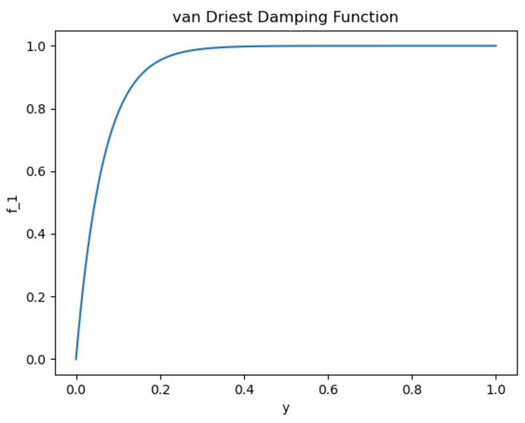


Figure 5.10: Van Driest damping function

5.1.1. Including wall-damping function

It is possible that the region near the wall has a lot of influence on the model we found. For instance, we see in figure 5.5 that the diagonal anisotropy stresses b_{uu}, b_{vv} and b_{ww} have a high peak near the wall. Wallin and Johansson suggested to weight the near-wall region by using a damping function, which damps the values at the wall making them less dominant [35]. For this we can use the van Driest function

$$f_1 = 1 - \exp -\frac{y^+}{A}, \tag{5.4}$$

where y^+ is the dimensionless wall distance which can be calculated with $y^+ = \frac{u_\tau y}{\nu}$ and A is a constant with a value of 26. The van Driest function against the normal wall distance is plotted in figure 5.10.

Wallin and Johansson aimed to develop an explicit algebraic Reynolds stress turbulence model. As basis they use the theorem of Pope and Caley-Hamilton to describe the anisotropic stress tensor as a combination of the ten invariant tensors in table 4.1. For a two-dimensional solution, they end up with the following model for the anisotropy including the near-wall van Driest damping function:

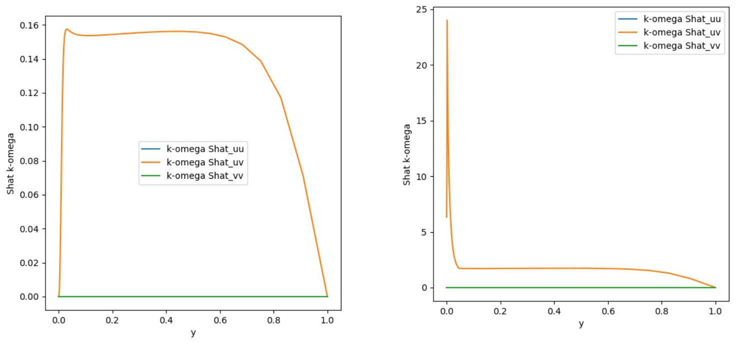
$$\boldsymbol{b} = f_1\beta_1 T^{(1)} + \left(f_1^2\beta_2 - (1-f_2^2)\frac{B_2}{2\max(\theta_1, \theta_1^{eq})}\right)T^{(2)} + (1-f_1^2)\frac{3B_2-4}{\max(\theta_1\theta_1^{eq})}T^{(3)}, \tag{5.5}$$

where B_2 is a constant, which could be evaluated using data. In [35], the authors chose $B_2 = 1.8$ since this choice yields a good fit to the DNS data. Equation 5.5 suggest to also include the candidate functions $f_1, f_1^2, (1-f_1)$ and $1-f_1^2$ in the library.

Until now, the shear and rotation stress tensors are scaled with the turbulence timescale $\tau = \frac{1}{\omega}$. However, this causes the scaled strain rate tensor to become zero near the wall. This could suggest that $\frac{\partial u}{\partial y}$ or S_{ij} is zero at the wall, but that is definitely not the case. A more appropriate expression for the time scale reads

$$\tau = \max\left(\frac{1}{\omega}, C_t\sqrt{\frac{\nu}{k\omega}}\right). \tag{5.6}$$

However, this timescale does not produce good results for $\hat{S}_{ij}$, which can be seen in Figure 5.11. A possible explanation for this is the use of near-wall-function used in the $k-\omega$ RANS simulation.



(a) $\hat{S}_{ij}$ with $\tau = \frac{1}{\omega}$

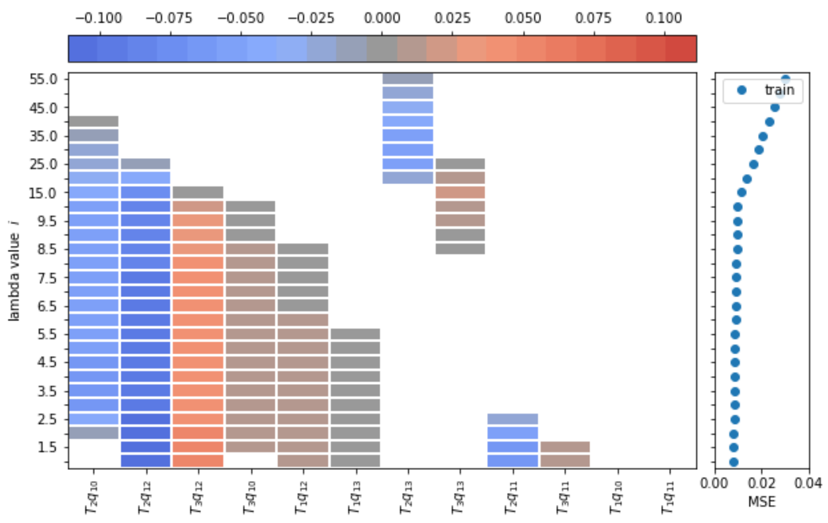
(b) $\hat{S}_{ij}$ with $\tau = \max\left(\frac{1}{\omega}, C_t\sqrt{\frac{\nu}{k\omega}}\right)$

Figure 5.11: $\hat{S}_{ij}$ with two different scaling factors.

Thus, we continue with the timescale $\tau = \frac{1}{\omega}$ and use the library

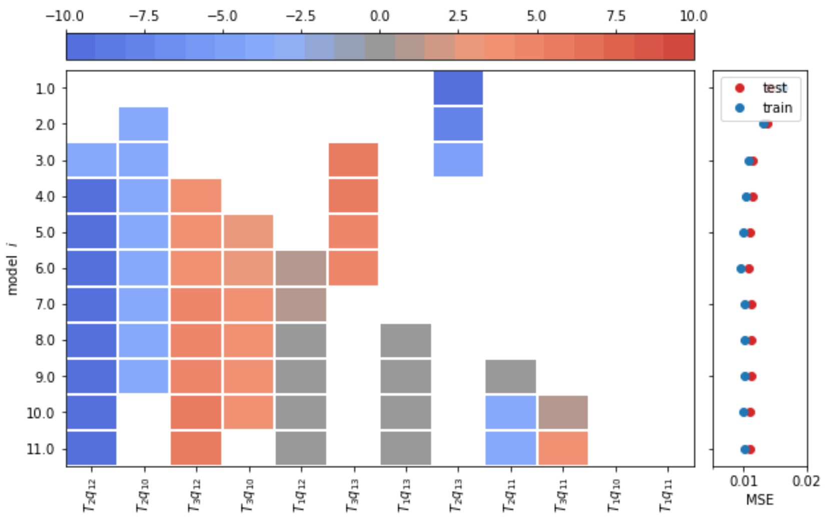
$$\begin{aligned}\mathcal{C} =&[f_1 T_{ij}^{(1)}, f_1 T_{ij}^{(2)}, f_1 T_{ij}^{(3)}, f_1^2 T_{ij}^{(1)}, f_1^2 T_{ij}^{(2)}, f_1^2 T_{ij}^{(3)},\\ &(1-f_1)T_{ij}^{(1)}, (1-f_1)T_{ij}^{(2)}, (1-f_1)T_{ij}^{(3)}, (1-f_1^2)T_{ij}^{(1)}, (1-f_1^2)T_{ij}^{(2)}, (1-f_1^2)T_{ij}^{(3)}].\end{aligned} \tag{5.7}$$

Model selection is done with the Lasso regression function, using values of λ between 1 and 100. Figure 5.12a shows the models for each λ. The model inference step is done with ridge regression using λ_r of 0.1 resulting in the models shown in figure 5.12b.



/

(a) Model selection



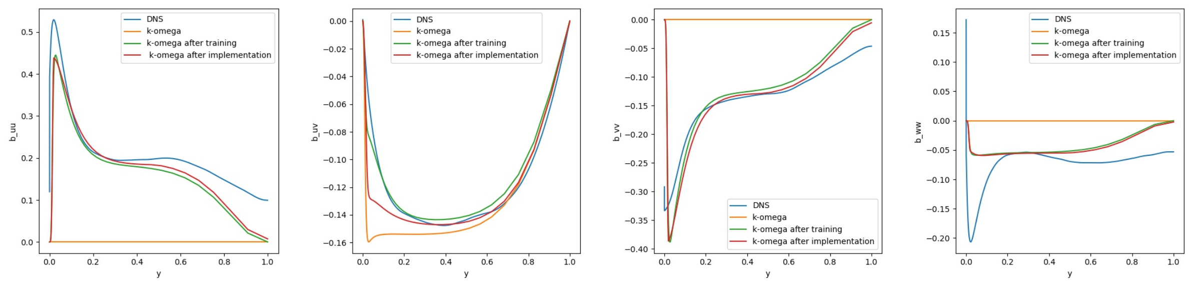
(b) Model inference

Figure 5.12: Sparse Symbolic Regression for the turbulent channel flow

The model with the lowest test error is

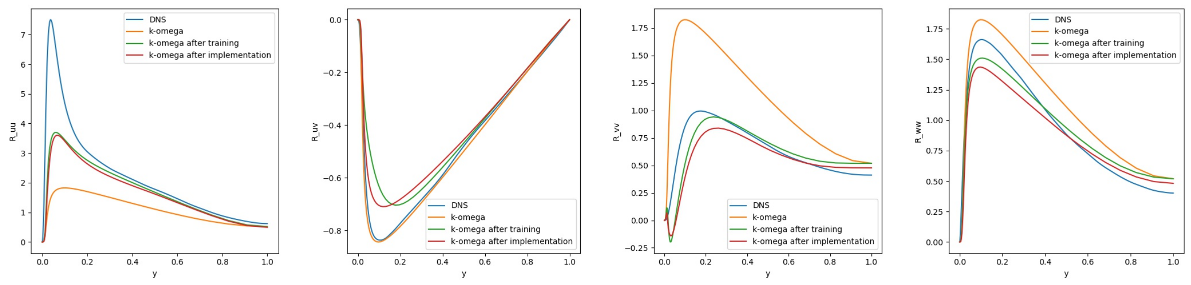
$$b^*_{ij} = 0.980(1-f_1^2)T_1 - 3.936 f_1 T_2 - 9.899(1-f_1)T_2 + 3.246 f_1 T_3 + 3.534(1-f_1)T_3 + 4.790(1-f_1^2)T_3. \quad (5.8)$$

The obtained model results in the corrected anisotropic stresses plotted in Figure 5.13. In this figure, also the results of the anisotropic Reynolds stress after propagation in OpenFoam are shown. The same results of the produced Reynolds stresses are shown in Figure 5.14. Figure 5.15 shows the velocity profiles of the propagated model. In these figures we can see that in the region near the wall the model better approximates the DNS data. So the near-wall damping function can have be important in modeling the anisotropic stresses. However, we also see that the Reynolds stress R_{vv} becomes negative, which is not realisable. Further research in the use of constraints in our optimisation functions should indicate if this can be circumvented.



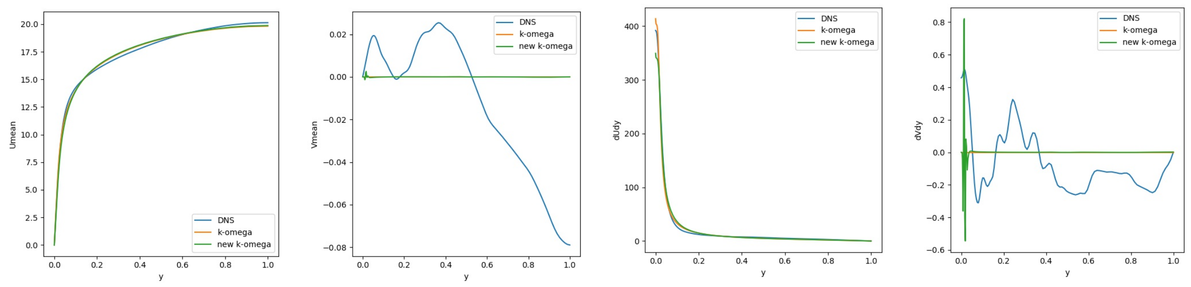
(a) b_{uu} **(b)** b_{uv} **(c)** b_{vv} **(d)** b_{ww}

Figure 5.13: Anisotropic part of Reynolds stress profiles of the DNS and RANS simulations and the obtained expression of a turbulent channel flow with $Re_\tau = 395$ using an expanded library containing the van Driest damping function.



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

Figure 5.14: Reynold stress profiles of the DNS and RANS simulations and the corrected expression of a turbulent channel flow with $Re_\tau = 395$ using an expanded library containing the van Driest damping function.



(a) U **(b)** mean V **(c)** $\frac{\partial U}{\partial y}$ **(d)** $\frac{\partial V}{\partial y}$

Figure 5.15: Velocity profiles of the DNS, RANS and a with wall-damping function corrected RANS simulation of a turbulent channel flow with $Re_\tau = 395$.

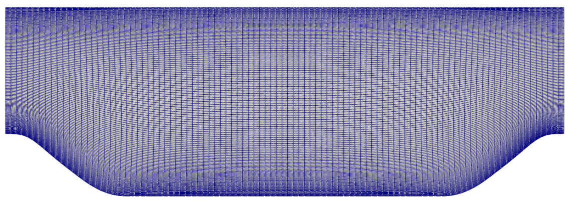
Table 5.4: Boundary and initial conditions used in the k-omega RANS simulation of the periodic hill

	Inlet	Outlet	Walls	Sides	Internal field
U	Cyclic	Cyclic	No Slip	empty	0.028
p	Cyclic	Cyclic	zeroGradient	empty	0
k	Cyclic	Cyclic	kLowReWallFunction	empty	$6e^{-8}$
ω	Cyclic	Cyclic	omegaWallFunction	empty	$1.5e^{-5}$
ν_τ	Cyclic	Cyclic	nutLowReWallFunction	empty	0.0

5.2. Periodic Hill

A more challenging case is the turbulent flow through a channel with periodic hills. The periodic hills cause the flow to separate, which is challenging for RANS models to capture accurately. Highly accurate data is provided by Xiao *et al.* [40], who performed DNS of flow over series of periodic hills at Reynolds number $Re = 5600$. Cyclic boundary conditions are applied in the streamwise (x) direction, and no-slip boundary conditions are applied at the top and bottom walls. The spanwise (z) 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 x-y planes. Fig 5.16 shows a schematic of the flow geometry and RANS-predicted velocity contour. The structured mesh consists of 100 and 150 cells in the x and y direction, respectively. As baseline RANS simulation we used the $k - \omega$ model with the boundary and initial conditions as indicated in table 5.4.


(a) Model selection for the PH with

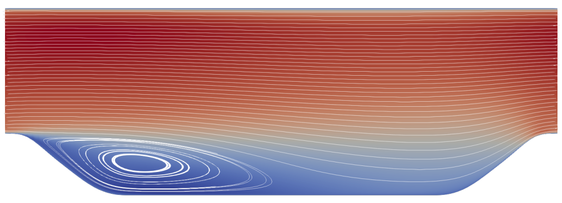

(b) Model inference

Figure 5.16: Structured hexahedral mesh and velocity field obtained from RANS simulation of the periodic hill, $Re = 5600$

5.2.1. Model Discovery

To give a preview of how the results will look like for this case, we start simple with a library only containing the single invariants I_1, I_2 and physical input features q_1 up till q_{10},

$$\Theta = [I_1, I_2, q_1, ...q_7]. \tag{5.9}$$

So now, there are no interactions between the features or other transformations. Each candidate function will be multiplied with basis tensors T_1, T_2 and T_3. The data will be divided into a training and test data set contain 75% and 25 % of the data respectively.

For these initial results we select the candidate function by performing a Lasso regression with values for the regularisation parameter $\lambda = [10e^{-4}, ..., 1]$ uniformly spaced using a log-scale. The sklearn package for python is used for the implementation of the lasso regression which relies on a coordinate descent method to solve the regression function. Since the values of the invariants I_1 and I_2 are approximately ten times smaller then all the physical parameters, we first apply a normalisation on the library to make the selection procedure independent of the magnitude of the candidates.

Figure 5.17 shows the result of the model selection using library 5.9. Each row in the graph shows the outcome of the lasso regression using the regularisation parameter as indicated on the y-axis. The colours indicate the value of the coefficients of the selected candidate functions, which are shown on x-axis indicates. The candidate functions on the x-axis are sorted on the amount of times it has been selected. So the graph may suggest that the candidate functions T_2, T_1, $T_1 q_1$ and $T_3 I_2$ are important, since they are selected most often.

With the unique combination of selected candidate functions an additional ridge regression is applied to inference the models. Now, the library is not normalised beforehand. As regularisation parameter $\lambda_r = 0.1$ is used. The results of the ridge regression are shown in figure 5.18. Similar as before, each row in this graph indicates a different model. Also visible are the errors between the target b^{Δ} and the outcome of the discovered model on the training and test data. These errors are better visible in figure 5.19, where they are plotted against the models' complexity, the amount of used candidate functions. The challenge is to select a model, which optimally balance error and complexity and is not over-fitting the data. We first select a model with the lowest error but a complexity of less than 10:

$$\begin{aligned}\mathcal{M}_1 =&(-0.86q_1 + 0.235)T_1\\ &+(3.145q_2 + 1.022q_5 - 5.115)T_2\\ &+(-1.829l_2 + 1.586q_1 - 0.514q_2)T_3\end{aligned} \tag{5.10}$$

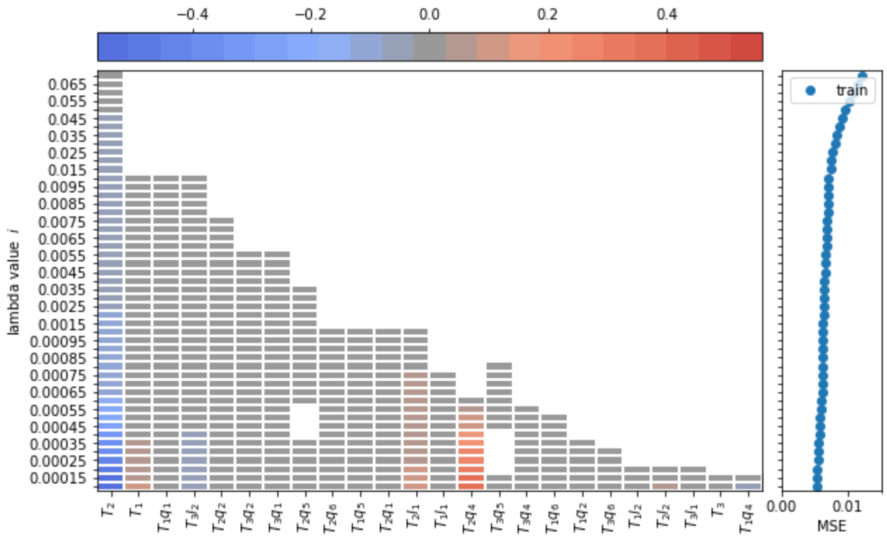


Figure 5.17: Model selection

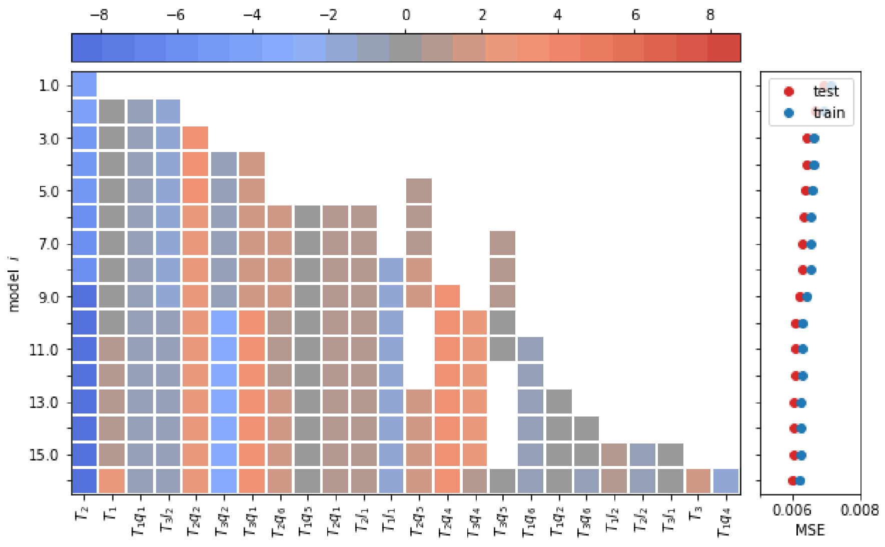


Figure 5.18: Model inference

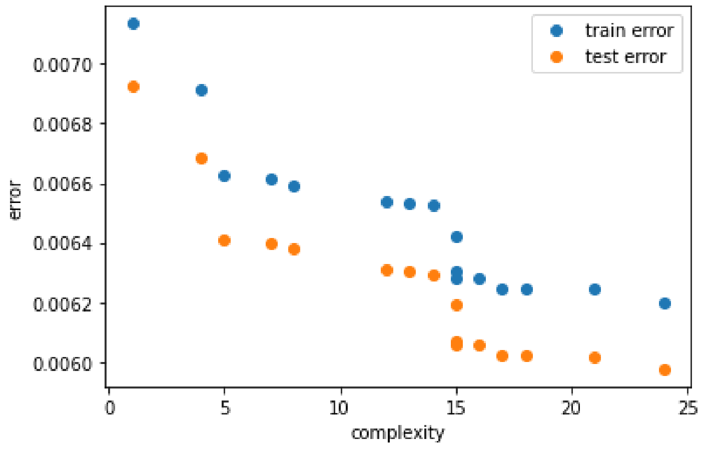


Figure 5.19: Train and test errors of discovered models

The model predicts the anisotropic stresses as shown in figure 5.20. At different cross sections through the channel, the stresses are plotted along the vertical distance. In this figure also the anisotropic stresses as modeled by the DNS and $k-\omega$ RANS simulation are plotted. Figure 5.21 shows the Reynolds stresses for each of the models. The model especially captures the anisotropic and Reynolds stress in the uu-direction better than the $k-\omega$ model. In the other direction, the outcome of our model and $k-\omega$ model are similar and in the vv-direction our models seems to perform even worse than the $k-\omega$ model. Finally, the model is implemented in OpenFOAM to obtain the velocity profiles. The results of these are shown in figure 5.22. Unfortunately, it looks like the velocity profiles are more off than the profiles of the $k-\omega$ model. The same can be observed for the kinetic energy profiles as plotted in figure 5.23. Hence the model does not perform as we wish for.

For comparison reasons we also select the most simple model:

$$\mathcal{M}_{\in} = -4.311T_2. \tag{5.11}$$

The same results are generated as the more complex model. For both models the mean squared errors of the anisotropic and Reynolds stresses are obtained, as well as the error of the streamwise velocity. The errors are stated in table 5.5. We see that both models improve the modeling of the anisotropic and Reynolds stresses compared to the baseline $k-\omega$ simulation. Model 1 has the lowest modeling error for the anisotropic stresses while model 2 has the lowest error for the Reynolds stresses. However, model 1 does not improve the streamwise velocity, since the MSE is higher than the $k-\omega$ model. The simple model does slightly improve the velocity profile and therefore, in this case, the simple model is more favourable than the more complex model.

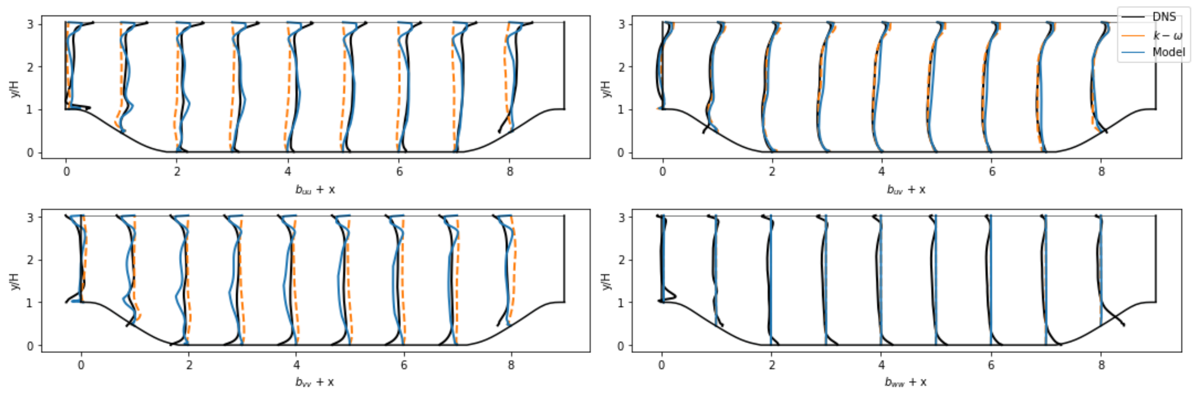


Figure 5.20: Anisotropic stresses b_{uu}, b_{uv}, b_{vv} and b_{ww} of the discovered model compared with DNS and $k-\omega$ RANS simulation

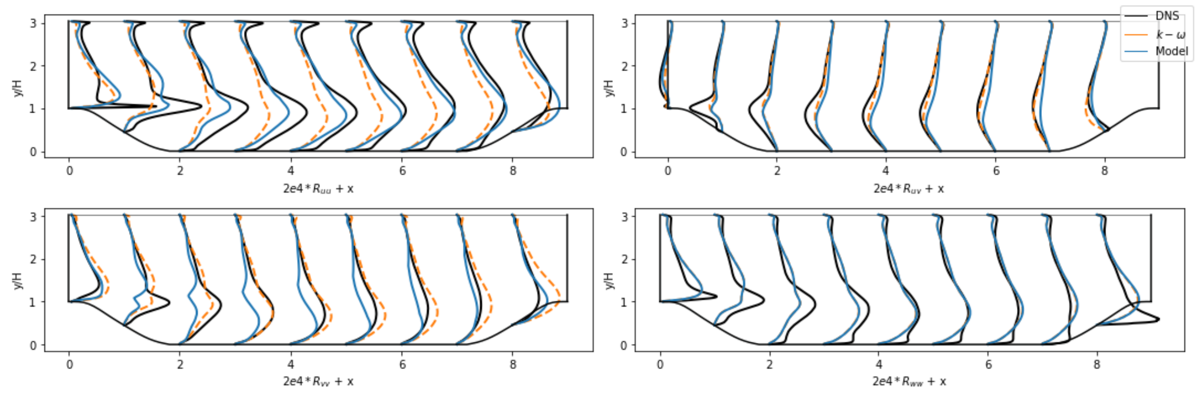


Figure 5.21: Reynolds stresses R_{uu}, R_{uv}, R_{vv} and R_{ww} of the discovered model compared with DNS and $k-\omega$ RANS simulation

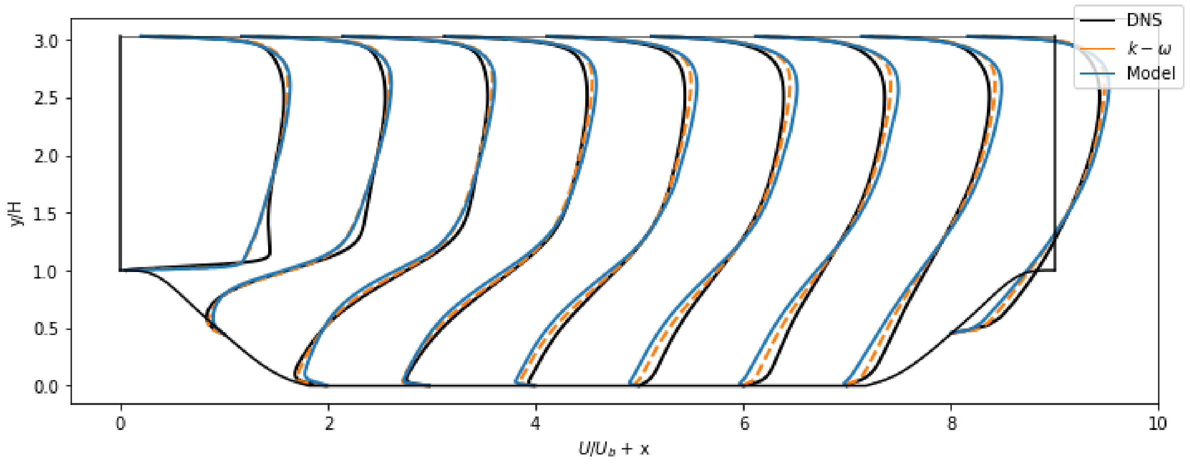


Figure 5.22: Velocity profile in streamwise direction of the discovered model compared with DNS and $k-\omega$ RANS simulation

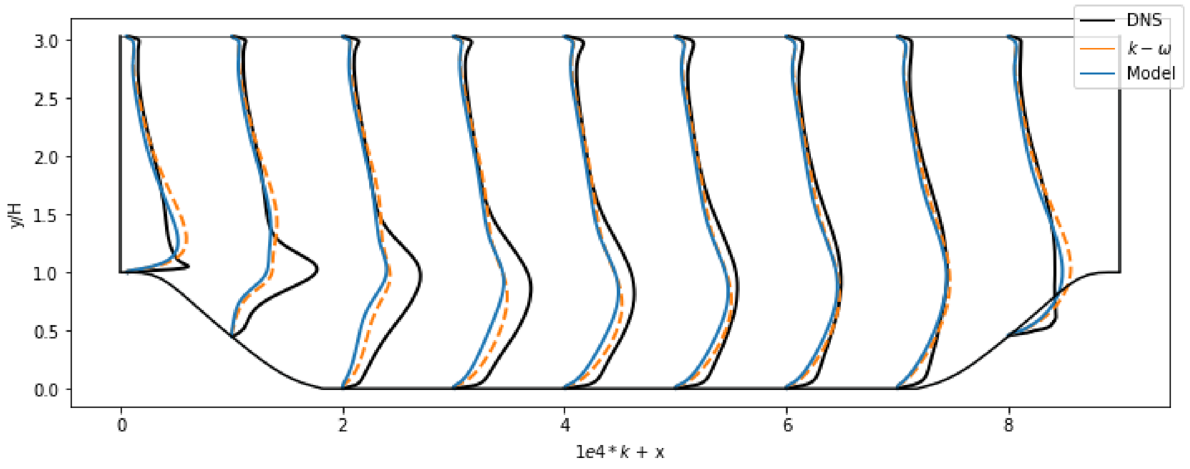


Figure 5.23: Kinetic energy profiles of the discovered model compared with DNS and $k-\omega$ RANS simulation

Table 5.5: MSE errors of the Anisotropic and Reynolds stress and the streamwise velocity for the two discovered models and the $k - \omega$ model.

	RANS	**Model 1**	**Model 2**
MSE (b_{ij})	0.012343	0.006537966	0.007096307
MSE (R_{ij})	5.21e-11	4.80047e-11	4.31072E-11
MSE (U)	1.91e-06	3.92945e-06	1.80531E-06

6 Intermediate conclusions and further research

So far, we have created a methodology to obtain models for the nonlinear anisotropic stresses using sparse symbolic regression. We also managed to implement the obtained models into the CFD solver OpenFoam to calculate the velocity profiles. Hence, a full model discovery pipeline is developed for discovering and testing data-driven turbulence models. In the rest of this project we want to focus on the following. First, we want to do more research in applying constraints to the optimisation functions. As we saw for the channel flow when adding the damping function, the Reynolds stresses became negative which should not be possible. Physical constraints could prevent this. Secondly, the influence of the near-wall damping function will be more investigated to find out what the best method is to include this function. One possibility is to include the function in the library of candidate function as we already did for the channel flow. Or we can apply the function later during the propagation of the model in Openfoam. Thereafter, we also would get more insight in the different optimisation functions which can be used in sparse symbolic regression. The optimisation problems we will consider are Lasso, Elasticnet, sr3 [44] and sequential thresholding of the least squares problem. The question is if these different regression methods result in different selections of the candidate function and different models. At last, we want to test the methodology on different test cases including the square duct flow, the Taylor-Couette flow and ultimately on a so called ’vortex gripper’, which serves as an industrial problem. In line with this, models will be trained on multiple cases simultaneously, to see if this contribute to a more general model.

References

[1] S Banerjee et al. “Presentation of anisotropy properties of turbulence, invariants versus eigenvalue approaches”. In: *Journal of Turbulence* 8.32 (2007). DOI: 10.1080/14685240701506896. URL: https://www.tandfonline.com/action/journalInformation?journalCode=tjot20.

[2] Andrea Beck, David Flad, and Claus Dieter Munz. “Deep neural networks for data-driven LES closure models”. In: *Journal of Computational Physics* 398 (Dec. 2019). ISSN: 10902716. DOI: 10.1016/j.jcp.2019.108910.

[3] Andrea Beck and Marius Kurz. “A perspective on machine learning methods in turbulence modeling”. In: *GAMM Mitteilungen* 44.1 (Mar. 2021). ISSN: 09367195. DOI: 10.1002/gamm.202100002.

[4] S. Beetham and J. Capecelatro. “Formulating turbulence closures using sparse regression with embedded form invariance”. In: *Physical Review Fluids* 5.8 (Aug. 2020). ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.5.084611.

[5] Steven L. Brunton. “Applying Machine Learning to Study Fluid Mechanics”. In: (Oct. 2021). URL: http://arxiv.org/abs/2110.02083.

[6] Steven L. Brunton et al. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences of the United States of America* 113.15 (2016), pp. 3932–3937. ISSN: 10916490. DOI: 10.1073/pnas.1517384113.

[7] KWING-SO CHOI and JOHN L. LUMLEY. “The return to isotropy of homogeneous turbulence”. In: *Journal of Fluid Mechanics* 436 (June 2001), pp. 59–84. ISSN: 0022-1120. DOI: 10.1017/S002211200100386X. URL: https://www.cambridge.org/core/product/identifier/S002211200100386X/type/journal_article.

[8] Steven Diamond and Stephen Boyd. *CVXPY: A Python-Embedded Modeling Language for Convex Optimization*. Tech. rep. 2016, pp. 1–5. URL: http://www.cvxpy.org/.

[9] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. “Annual Review of Fluid Mechanics: Turbulence Modeling in the Age of Data”. In: *Annu. Rev. Fluid Mech* 51 (2019), pp. 357–377. DOI: 10.1146/annurev-fluid-010518. URL: https://doi.org/10.1146/annurev-fluid-010518-.

[10] Bernhard Eisfeld, Chris Rumsey, and Vamshi Togiti. “Verification and validation of a second-moment-closure model”. In: *AIAA Journal*. Vol. 54. 5. American Institute of Aeronautics and Astronautics Inc., 2016, pp. 1524–1541. DOI: 10.2514/1.J054718.

[11] Hamidreza Eivazi et al. “Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations”. In: (July 2021). URL: http://arxiv.org/abs/2107.10711.

[12] Masataka Gamahara and Yuji Hattori. “Searching for turbulence models by artificial neural network”. In: (June 2016). DOI: 10.1103/PhysRevFluids.2.054604. URL: http://arxiv.org/abs/1607.01042%20http://dx.doi.org/10.1103/PhysRevFluids.2.054604.

[13] T. B. Gatski and C. G. Speziale. “On Explicit Algebraic Stress Models for Complex Turbulent Flows”. In: *Journal of Fluid Mechanics* 254 (1993), pp. 59–78. ISSN: 14697645. DOI: 10.1017/S0022112093002034.

[14] Tin Kam Ho. “Random decision forests”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. Vol. 1. IEEE Computer Society, 1995, pp. 278–282. ISBN: 0818671289. DOI: 10.1109/ICDAR.1995.598994.

[15] Mikael L.A. Kaandorp and Richard P. Dwight. “Data-driven modelling of the Reynolds stress tensor using random forests with invariance”. In: *Computers and Fluids* 202 (Apr. 2020). ISSN: 00457930. DOI: 10.1016/j.compfluid.2020.104497.

[16] Teuvo Kohonen. *An Introduction to Neural Computing*. Tech. rep. 1988, pp. 3–16.

[17] J. Ling and J. Templeton. “Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty”. In: *Physics of Fluids* 27.8 (Aug. 2015). ISSN: 10897666. DOI: 10.1063/1.4927765.

[18] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In: *Journal of Fluid Mechanics* 807 (Nov. 2016), pp. 155–166. ISSN: 14697645. DOI: 10.1017/jfm.2016.615.

[19] John L. Lumley. “Computational Modeling of Turbulent Flows”. In: *Advances in Applied Mechanics* 18.C (Jan. 1979), pp. 123–176. ISSN: 0065-2156. DOI: 10.1016/S0065-2156(08)70266-7.

[20] John L. Lumley and Gary R. Newman. “The return to isotropy of homogeneous turbulence”. In: *Journal of Fluid Mechanics* 82.1 (Aug. 1977), pp. 161–178. ISSN: 0022-1120. DOI: 10.1017/S0022112077000585. URL: https://www.cambridge.org/core/product/identifier/S0022112077000585/type/journal_article.

[21] Shirui Luo et al. *Review and Examination of Input Feature Preparation Methods and Machine Learning Models for Turbulence Modeling*. Tech. rep.

[22] R. Maulik and O. San. “A neural network approach for the blind deconvolution of turbulent flows”. In: *Journal of Fluid Mechanics* 831 (Nov. 2017), pp. 151–181. ISSN: 14697645. DOI: 10.1017/jfm.2017.637.

[23] Robert D. Moser, John Kim, and Nagi N. Mansour. “Direct numerical simulation of turbulent channel flow up to Re$_T$=590”. In: *Physics of Fluids* 11.4 (1999), pp. 943–945. ISSN: 10706631. DOI: 10.1063/1.869966.

[24] Frans T M Nieuwstadt et al. *Turbulence Introduction to Theory and Applications of Turbulent Flows*. Tech. rep.

[25] S. B. Pope. “A more general effective-viscosity hypothesis”. In: *Journal of Fluid Mechanics* 72.02 (Nov. 1975), p. 331. ISSN: 0022-1120. DOI: 10.1017/S0022112075003382. URL: http://www.journals.cambridge.org/abstract_S0022112075003382.

[26] 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”. In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.10.045.

[27] Maximilian Reissmann et al. “Application of Gene Expression Programming to a-posteriori LES modeling of a Taylor Green Vortex”. In: *Journal of Computational Physics* 424 (Jan. 2021). ISSN: 10902716. DOI: 10.1016/j.jcp.2020.109859.

[28] Sal Rodriguez and Practical Tools. *Applied Computational Fluid Dynamics and Turbulence Modeling*.

[29] A L Samuel. *Some studies in machine learning using the game of checkers*. Tech. rep.

[30] Martin Schmelzer, Richard P. Dwight, and Paola Cinnella. “Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression”. In: *Flow, Turbulence and Combustion* 104.2-3 (Mar. 2020), pp. 579–603. ISSN: 15731987. DOI: 10.1007/s10494-019-00089-x.

[31] Alexander J Smits. *Lectures in Fluid Mechanics Viscous Flows and Turbulence*. Tech. rep. 2009.

[32] Salar Taghizadeh et al. “Turbulence closure modeling with data-driven techniques: Investigation of generalizable deep neural networks”. In: (Feb. 2021). URL: http://arxiv.org/abs/2102.01487.

[33] Brendan Tracey, Karthik Duraisamy, and Juan 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. ISBN: 9781624103438. DOI: 10.2514/6.2015-1287.

[34] A. Vollant, G. Balarac, and C. Corre. “Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures”. In: *Journal of Turbulence* 18.9 (Sept. 2017), pp. 854–878. ISSN: 14685248. DOI: 10.1080/14685248.2017.1334907.

[35] Stefan Wallin and Arne V. Johansson. “An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows”. In: *Journal of Fluid Mechanics* 403 (Jan. 2000), pp. 89–132. ISSN: 00221120. DOI: `10.1017/S0022112099007004`.

[36] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. “A Physics Informed Machine Learning Approach for Reconstructing Reynolds Stress Modeling Discrepancies Based on DNS Data”. In: (June 2016). DOI: `10.1103/PhysRevFluids.2.034603`. URL: `http://arxiv.org/abs/1606.07987%20http://dx.doi.org/10.1103/PhysRevFluids.2.034603`.

[37] J. Weatheritt and R. D. Sandberg. “The development of algebraic stress models using a novel evolutionary algorithm”. In: *International Journal of Heat and Fluid Flow* 68 (Dec. 2017), pp. 298–318. ISSN: 0142727X. DOI: `10.1016/j.ijheatfluidflow.2017.09.017`.

[38] Jin Long Wu, Heng Xiao, and Eric Paterson. “Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework”. In: *Physical Review Fluids* 7.3 (July 2018). ISSN: 2469990X. DOI: `10.1103/PhysRevFluids.3.074602`.

[39] Jin-Long Wu et al. “RANS Equations with Explicit Data-Driven Reynolds Stress Closure Can Be Ill-Conditioned”. In: (Mar. 2018). DOI: `10.1017/jfm.2019.205`. URL: `http://arxiv.org/abs/1803.05581%20http://dx.doi.org/10.1017/jfm.2019.205`.

[40] Heng Xiao et al. “Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations”. In: *Computers & Fluids* 200 (Mar. 2020), p. 104431. ISSN: 0045-7930. DOI: `10.1016/J.COMPFLUID.2020.104431`.

[41] Chenyue Xie et al. “Modeling subgrid-scale force and divergence of heat flux of compressible isotropic turbulence by artificial neural network”. In: *Physical Review Fluids* 4.10 (Oct. 2019). ISSN: 2469990X. DOI: `10.1103/PhysRevFluids.4.104605`.

[42] Yuhui Yin et al. “Feature selection and processing of turbulence modeling based on an artificial neural network”. In: *Physics of Fluids* 32.10 (Oct. 2020). ISSN: 10897666. DOI: `10.1063/5.0022561`.

[43] Yaomin Zhao et al. “RANS turbulence model development using CFD-driven machine learning”. In: *Journal of Computational Physics* 411 (June 2020). ISSN: 10902716. DOI: `10.1016/j.jcp.2020.109413`.

[44] Peng Zheng et al. “A Unified Framework for Sparse Relaxed Regularized Regression: SR3”. In: *IEEE* 7 (2019), pp. 1404–1423. DOI: `10.1109/ACCESS.2018.2886528`. URL: `http://www.ieee.org/publications_standards/publications/rights/index.html`.

[45] Zhi-Hua Zhou. *Machine Learning*. Singapore: Springer Singapore, 2021. ISBN: 978-981-15-1966-6. DOI: `10.1007/978-981-15-1967-3`. URL: `https://link.springer.com/10.1007/978-981-15-1967-3`.