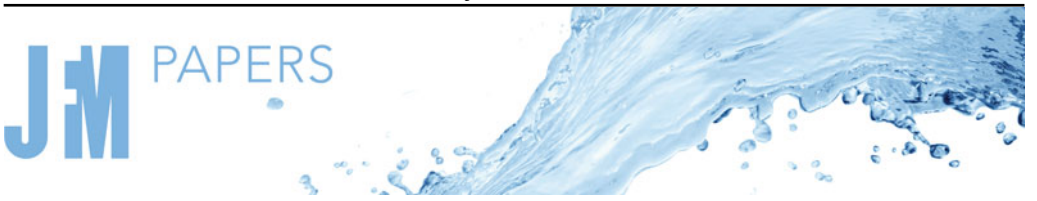
J. Fluid Mech. (2025), *vol.* 1019, A49, doi:10.1017/jfm.2025.10618



Realisability-informed machine learning for turbulence anisotropy mappings

Ryley McConkey[1,2], **Nikhila Kalia**[2], **Eugene Yee**[2] **and Fue-Sang Lien**[2]

[1]Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA

[2]University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1, Canada

Corresponding author: Ryley McConkey, rmcconke@mit.edu

(Received 20 December 2024; revised 3 June 2025; accepted 1 August 2025)

Within the context of machine learning-based closure mappings for Reynolds-averaged Navier Stokes turbulence modelling, physical realisability is often enforced using *ad hoc* postprocessing of the predicted anisotropy tensor. In this study, we address the realisability issue via a new physics-based loss function that penalises non-realisable results during training, thereby embedding a preference for realisable predictions into the model. Additionally, we propose a new framework for data-driven turbulence modelling which retains the stability and conditioning of optimal eddy viscosity-based approaches while embedding equivariance. Several modifications to the tensor basis neural network to enhance training and testing stability are proposed. We demonstrate the conditioning, stability and generalisation of the new framework and model architecture on three flows: flow over a flat plate, flow over periodic hills and flow through a square duct. The realisability-informed loss function is demonstrated to significantly increase the number of realisable predictions made by the model when generalising to a new flow configuration. Altogether, the proposed framework enables the training of stable and equivariant anisotropy mappings, with more physically realisable predictions on new data. We make our code available for use and modification by others. Moreover, as part of this study, we explore the applicability of Kolmogorov–Arnold networks to turbulence modelling, assessing its potential to address nonlinear mappings in the anisotropy tensor predictions and demonstrating promising results for the flat plate case.

Key words: machine learning, turbulence modelling, turbulence simulation

It is prohibitively expensive to resolve all relevant scales of turbulence for industrially relevant flows. Even with increasing computational capacity, Kolmogorov microscale-resolving techniques such as direct numerical simulation (DNS) will be out of reach for

decades (Slotnick *et al.* 2014). In order to enable the practical simulation of turbulent flows, a variety of techniques are currently in use, such as Reynolds-averaged Navier Stokes (RANS), detached eddy simulation (DES) and large eddy simulation (LES). Each technique comes with its own advantages. The user's available computational resources are the primary consideration. The RANS technique, which models turbulence as a single-scale phenomenon, remains the most popular industrial technique due to the computational costs of scale-resolving simulations (Witherden & Jameson 2017).

In recent years, a new menu item for industrial turbulence modelling is emerging. Machine learning has been used to augment RANS, DES, LES and DNS, with various objectives, such as accelerating simulation times (Kochkov *et al.* 2021), inferring optimal coefficient fields (Singh, Duraisamy & Zhang 2017), model calibration (Matai & Durbin 2019) and turbulence model augmentation (Duraisamy, Iaccarino & Xiao 2019; Brunton, Noack & Koumoutsakos 2020). For LES, emerging research on training *in situ* and within a differentiable solver offers the prospect of a generalisable subgrid model (List, Chen & Thuerey 2022; Sirignano & MacArt 2023). It is clear that for LES, allowing the model to interact with a solver for an unsteady flow vastly improves the generalisability of the learned closure relationship (List *et al.* 2022). Within the context of steady-state RANS simulations, a promising new menu item is the ability to train a specialised turbulence model with a given industrial dataset. Significant attention has been given to the development of machine learning-augmented closure models. Specifically, flow-specific sensitisation of the stable but often inaccurate linear eddy viscosity relationship via machine learning is an area of major interest. In their seminal work, Ling, Kurzawski & Templeton (2016) proposed a neural network architecture based on a tensor basis expansion of the anisotropy tensor, referred to as a tensor basis neural network (TBNN). This architecture was extended to random forests by Kaandorp (2018) and Kaandorp & Dwight (2020). The TBNN architecture has been used in several studies, such as by Song *et al.* (2019) and Zhang *et al.* (2019). Further modifications to the TBNN framework within the context of simple channel flows have been proposed by Cai *et al.* (2022, 2024). Specifically, Cai *et al.* studied issues related to non-unique mappings between the closure term and input features for plane channel flow, an issue reported by Liu *et al.* (2021). This issue primarily occurs when a small input feature set is used, such as the 5 invariants (several of which are zero for two-dimensional flows) used in Ling *et al.*'s original TBNN (Ling *et al.* 2016). Several strategies have been proposed to address the issue of a non-unique mapping, including incorporation of additional input features by Wu, Xiao & Paterson (2018), and ensembling through a divide and conquer approach by Man *et al.* (2023). In the present investigation, we address the issue related to non-uniqueness of the mapping via use of a rich input feature set. In an effort to make these mappings more transparent, Mandler & Weigand (2023) analysed predictions made via an anisotropy mapping using input feature importance metrics such as Shapley additive explanation values. Interpretability analysis sheds light on which RANS features contain more information about different flow physics, which helps guide future input feature selection (Mandler & Weigand 2023). The TBNN-type models are not the only architecture in use – others include the eigenvalue reconstruction technique proposed by Wu *et al.* (2018, 2019*a*), and the Reynolds force vector approach proposed by Cruz *et al.* (2019), and further investigated by Brener *et al.* (2022) and Amarloo, Forooghi & Abkar (2022).

Here, we consider the specific problem of training data-driven anisotropy mappings between RANS and a higher fidelity closure term from LES or DNS. Several open questions remain in the area of machine learning anisotropy mappings for RANS, with perhaps the most popular question being whether a 'universal turbulence model' could be produced via machine learning (Duraisamy 2021; Spalart 2023). However, when it comes

to generalisability of these mappings, the no-free-lunch theorem is at play as demonstrated in McConkey *et al.* (2022*b*). Accurate sensitisation of the anisotropy mapping for a given flow comes at the cost of generalisation to completely new flows. Nevertheless, numerous studies show that the sensitised mapping generalises well within the same flow type (Wu *et al.* 2018; Kaandorp & Dwight 2020; McConkey *et al.* 2022*b*; Man *et al.* 2023). While this lack of new flow generalisability means that a machine learning-augmented turbulence model is off the menu for some applications, it remains on the menu for many industrial applications. For example, an industrial user with LES data can leverage these techniques to develop an augmented RANS turbulence model sensitised to a flow of interest. Our opinion is that this lack of generalisability is due to a lack of sufficient data. Despite numerous datasets for this purpose being available, we believe the entire space of possible mappings between RANS input features and higher fidelity closure terms is still sparsely covered by available data. For the foreseeable future, these mappings will therefore be limited to flow-specific sensitisation.

While the generalisability question is perhaps the most popular one, several other key issues remain in training anisotropy mappings for RANS. How can predicted quantities be injected in a stable and well-conditioned manner? How can eddy viscosity-based approaches be united with TBNN-type approaches? How can we incorporate alternatives to the traditional fully connected layers in a TBNN? In this investigation, we address several of these remaining questions. We formulate an injection framework which unites TBNN-type model architectures with the more stable optimal eddy viscosity-based techniques. This unification enables stable use of the equivariance-enforcing TBNN within the momentum equation, without the use of stabilising blending factors (Kaandorp & Dwight 2020). The proposed framework also has the advantage of producing a well-conditioned solution without the use of an optimal eddy viscosity, an often-unstable quantity that is difficult to predict via a machine learning model. We also target the problem of producing realisable predictions via a TBNN-type architecture via a physics-based loss function penalty (*viz.*, incorporation of a learning bias in the framework). To further improve the flexibility and representational capacity of TBNN-type models, we investigate the inclusion of a Kolmogorov–Arnold network (KAN) (Liu *et al.* 2024) into the framework to replace the multi-layer perception in the TBNN. We also further investigate the invariant input feature sets commonly used for TBNN architectures, and provide new insights as to which input features are appropriate for use in flows with certain zero gradient directions. We demonstrate good generalisation performance of the 'realisability-informed' TBNN. We also demonstrate that the realisability-informed loss function greatly reduces non-realisable predictions when generalising.

The present work unites TBNN-type frameworks (for example Ling *et al.* 2016; Kaandorp 2018; Kaandorp & Dwight 2020; and Man *et al.* 2023) with optimal eddy viscosity frameworks (e.g. Wu *et al.* 2018; Brener *et al.* 2021; and McConkey *et al.* 2022*a*). The advantage here is maintaining a stable injection environment (Wu *et al.* 2019*b*; Brener *et al.* 2021), while also retaining the simplicity, elegance and implicit equivariance of the TBNN architecture. Additionally, the present work is the first to implement a way to inform the TBNN of physical realisability during the training process. Whereas most existing techniques to enforce realisability of the neural network involve an ad-hoc post-processing step, our technique leverages physical realisability as an additional training target, thereby embedding an additional physics-based (learning) bias into the model. This idea extends the learning bias proposed by Riccius, Agrawal & Koutsourelakis (2023). Lastly, despite widespread use of minimal integrity basis input features for two-dimensional flows, and flows through a square duct, the present investigation is the first to systematically examine these input features for flow through a square duct. This examination leads to several

unsuitable input features being identified, and a codebase for investigating other flows of interest.

This manuscript is organised as follows. Section 1 describes the novel techniques in detail, including the injection framework ((1.1)–(1.2)), realisability-informed loss function (1.3) and improved TBNN architecture (1.4). Details on the datasets, input features, hyperparameters, implementation and code availability are given in § 1.5. Two primary results are presented in § 2: generalisation tests of the realisability-informed TBNN (2.1), and an examination of how realisability-informed training produces more realisable predictions when generalising (2.2). Conclusions and future work are discussed in § 3.

1. Methodology

1.1. *Injecting Reynolds stress tensor decompositions used in data-driven closure modelling frameworks*

The steady-state, RANS momentum equations for an incompressible, Newtonian fluid are given by (Reynolds 1895)

$$U_i \frac{\partial U_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial P}{\partial x_j} + \nu \frac{\partial^2 U_j}{\partial x_i \partial x_i} - \frac{\partial \tau_{ij}}{\partial x_i}, \tag{1.1}$$

where U_j is the mean velocity, P is the mean pressure, ρ is the fluid density, ν is the kinematic viscosity and τ_{ij} is the Reynolds stress tensor.

The continuity equation also applies to this flow

$$\frac{\partial U_i}{\partial x_i} = 0. \tag{1.2}$$

Together, (1.1) and (1.2) are unclosed. In the most general case, there are four equations (continuity + 3 momentum), and 10 unknowns: P, 3 components of U_i and 6 components of τ_{ij}. The goal of turbulence closure modelling is to express τ_{ij} in terms of P and U_i.

The Reynolds stress tensor can be decomposed into isotropic (hydrostatic) and anisotropic (deviatoric) components

$$\tau_{ij} = \frac{2}{3}k\delta_{ij} + a_{ij}, \tag{1.3}$$

where $k = (1/2)\tau_{kk}$ is half the trace of the Reynolds stress tensor, and a_{ij} is the anisotropy tensor. The isotropic component ($(2/3)k\delta_{ij}$) in (1.3) can be absorbed into the pressure gradient term in (1.1) to form a modified pressure

$$U_i \frac{\partial U_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial}{\partial x_j}\left(P + \frac{2}{3}\rho k\right) + \nu \frac{\partial^2 U_j}{\partial x_i \partial x_i} - \frac{\partial a_{ij}}{\partial x_i}. \tag{1.4}$$

Equation (1.4) is the form of the RANS momentum equation commonly used in RANS turbulence closure modelling. Several data-driven closure modelling frameworks are based on modelling the Reynolds stress tensor itself, or it's divergence (as in Brener *et al.* (2022)), which imply the use of (1.1). In our work, we model the Reynolds stress anisotropy tensor a_{ij}, implying the use of (1.4). In either case, the term 'injection' refers to using a model prediction as the closure term in the momentum equation and numerically solving the momentum equation with this predicted closure term.

In an eddy viscosity hypothesis, the anisotropy tensor a_{ij} is postulated to be a function of the mean strain rate S_{ij} and rotation rate R_{ij} tensors

$$a_{ij} = a_{ij}(S_{ij},\, R_{ij}), \tag{1.5}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right), \tag{1.6}$$

$$R_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right). \tag{1.7}$$

The eddy viscosity hypothesis implies the following important constraints:

(i) The anisotropy tensor can be predicted locally and instantaneously. In (1.5), there is no temporal dependence.
(ii) The anisotropic turbulent stresses are caused entirely by mean velocity gradients.

There are many examples of turbulent flows where these assumptions are violated (Pope 2000). For example, history effects occur in boundary layers (Bobke *et al.* 2017). Nevertheless these approximations remain widely used in eddy viscosity modelling. The most common eddy viscosity hypothesis is the linear eddy viscosity hypothesis

$$a_{ij} = -2\nu_t S_{ij}, \tag{1.8}$$

where ν_t is the eddy viscosity, a scalar. This linear hypothesis draws direct analogy from the stress–strain rate relation for a Newtonian fluid. Along with the important constraints implied by invoking an eddy viscosity hypothesis, the linear eddy viscosity hypothesis implies the following:

(i) The anisotropy tensor is aligned with the mean strain rate tensor.
(ii) The mapping between mean strain rate and anisotropic stress is isotropic, in that it can be represented using a single scalar (ν_t).

More general nonlinear eddy viscosity hypotheses have been used in several models. The primary advantage of these models is that they permit a misalignment of the principal axes of a_{ij} and S_{ij}, which occurs for even simple flows. Applying Cayley–Hamilton theorem (Hamilton 1853; Cayley 1858) to (1.5), Pope (1975) derived the most general expression for a nonlinear eddy viscosity model

$$a_{ij} = 2k\sum_{n=1}^{10} g_n \hat{T}_{ij}^{(n)}, \tag{1.9}$$

where $n = 1, 2, \ldots 10$ indexes the scalar coefficients g_n, and the following basis tensors:

$$\begin{aligned}
\hat{T}_{ij}^{(1)} &= \hat{S}_{ij}, & \hat{T}_{ij}^{(6)} &= \hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lj} + \hat{S}_{ik}\hat{R}_{kl}\hat{R}_{lj} - \tfrac{2}{3}\hat{S}_{kl}\hat{R}_{lm}\hat{R}_{mk}\delta_{ij},\\
\hat{T}_{ij}^{(2)} &= \hat{S}_{ik}\hat{R}_{kj} - \hat{R}_{ik}\hat{S}_{kj}, & \hat{T}_{ij}^{(7)} &= \hat{R}_{ik}\hat{S}_{kl}\hat{R}_{lm}\hat{R}_{mj} - \hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{R}_{mj},\\
\hat{T}_{ij}^{(3)} &= \hat{S}_{ik}\hat{S}_{kj} - \tfrac{1}{3}\hat{S}_{kl}\hat{S}_{lk}\delta_{ij}, & \hat{T}_{ij}^{(8)} &= \hat{S}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{S}_{mj} - \hat{S}_{ik}\hat{S}_{kl}\hat{R}_{lm}\hat{S}_{mj},\\
\hat{T}_{ij}^{(4)} &= \hat{R}_{ik}\hat{R}_{kj} - \tfrac{1}{3}\hat{R}_{kl}\hat{R}_{lk}\delta_{ij}, & \hat{T}_{ij}^{(9)} &= \hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{S}_{mj} + \hat{S}_{ik}\hat{S}_{kl}\hat{R}_{lm}\hat{R}_{mj} - \tfrac{2}{3}\hat{S}_{kl}\hat{S}_{lm}\hat{R}_{mo}\hat{R}_{ok}\delta_{ij},\\
\hat{T}_{ij}^{(5)} &= \hat{R}_{ik}\hat{S}_{kl}\hat{S}_{lj} - \hat{S}_{ik}\hat{S}_{kl}\hat{R}_{lj}, & \hat{T}_{ij}^{(10)} &= \hat{R}_{ik}\hat{S}_{kl}\hat{S}_{lm}\hat{R}_{mo}\hat{R}_{oj} - \hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{S}_{mo}\hat{R}_{oj}\,.
\end{aligned} \tag{1.10}$$

The tensors $\hat{S}_{ij}$ and $\hat{R}_{ij}$ above are the non-dimensionalised strain rate and rotation rate tensors. In Pope's original work, these tensors are given by

$$\hat{S}_{ij} = \frac{k}{\varepsilon} S_{ij}, \tag{1.11}$$

$$\hat{R}_{ij} = \frac{k}{\varepsilon} R_{ij}. \tag{1.12}$$

However, other normalisation constants are possible, and in several nonlinear eddy viscosity models, different basis tensors use different normalisation constants. To our knowledge, varying the basis tensor normalisation has not been explored in data-driven turbulence closure modelling.

With Pope's general expression for the anisotropy tensor, (1.4) becomes

$$U_i \frac{\partial U_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial}{\partial x_j}\left(P + \frac{2}{3}\rho k\right) + \nu \frac{\partial^2 U_j}{\partial x_i \partial x_i} - \frac{\partial}{\partial x_i}\left(2k \sum_{n=1}^{10} g_n \hat{T}_{ij}^{(n)}\right). \tag{1.13}$$

This is the form of the momentum equation used in several studies which aim to augment the closure relationship via machine learning. For example, in Ling *et al.*'s TBNN investigation (Ling *et al.* 2016), (1.13) was used. Kaandorp (2018) and Kaandorp & Dwight (2020) also used this form of the momentum equation. The TBNN-based approaches are advantageous for several reasons. The TBNN architecture is motivated by a tensor algebra-based argument that if the anisotropy tensor is to be expanded in terms of the mean strain and rotation rate tensors (a dominant approach in eddy viscosity modelling), the TBNN is the most general form of this expansion (Pope 1975). Additionally, other techniques to incorporate equivariance are cumbersome, requiring learning and predicting in an invariant eigenframe of some tensorial quantity (Wu *et al.* 2018; Brener *et al.* 2022). The TBNN-based architectures do not require computing eigenvalues of the strain rate tensor at every evaluation data point. Lastly, this closure term is highly expressive, in that ten different combinations of S_{ij} and R_{ij} can be used to represent the anisotropy tensor. However, a major disadvantage with numerically solving (1.13) is that the closure term is entirely explicit, greatly reducing numerical stability and conditioning (Brener *et al.* 2021). For this reason, Kaandorp & Dwight (2020) needed to implement a blending function, which blends the fully explicit closure term in (1.13) with the more stable implicit closure term treatment made possible with a linear eddy viscosity hypothesis. Here, implicit treatment refers to the way the velocity discretisation matrix is constructed in the numerical solver. When a term is treated implicitly, it contributes to the U_i coefficient matrix. In contrast, explicit treatment refers to treating a term as a fixed source term in the discretised equation. Assuming the closure term takes the form $a_{ij} = -2\nu_t S_{ij}$, (1.4) can be written as

$$U_i \frac{\partial U_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial}{\partial x_j}\left(P + \frac{2}{3}\rho k\right) + \frac{\partial}{\partial x_i}\left[(\nu + \nu_t)\frac{\partial U_j}{\partial x_i}\right]. \tag{1.14}$$

Equation (1.14) has the major advantage of increasing diagonal dominance of the U_j coefficient matrix obtained from discretisation of this equation, via the eddy viscosity. However, this closure framework only permits the inaccurate linear eddy viscosity closure approximation.

In the present work, we propose the following hybrid treatment of the closure term:

$$U_i \frac{\partial U_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial}{\partial x_j}\left(P + \frac{2}{3}\rho k\right) + \frac{\partial}{\partial x_i}\left[(\nu + \nu_t)\frac{\partial U_j}{\partial x_i}\right] - \frac{\partial}{\partial x_i}\left(2k \sum_{n=2}^{10} g_n \hat{T}_{ij}^{(n)}\right). \tag{1.15}$$

In (1.15), the $n = 1$ (linear) term has been separated and receives implicit treatment, while the remaining $n = 2, 3, \ldots 10$ terms grant an opportunity for a machine learning model to provide rich representation of the nonlinear part of a_{ij}. As we will discuss, the separation of the linear term requires special treatment during training and closure term injection.

1.2. *Conditioning analysis*

Various decompositions of the Reynolds stress tensor were investigated by Brener *et al.*'s conditioning analysis (Brener *et al.* 2021). Conditioning analysis for data-driven turbulence closure frameworks is important, since ill-conditioned momentum equations have the potential to amplify errors in the predicted closure term. Brener *et al.* (2021) concluded that an optimal eddy viscosity approach is necessary to achieve a well-conditioned solution, since it incorporates information about the DNS mean velocity field. In the present work, we demonstrate that the following closure decomposition:

$$a_{ij} = -2\nu_t^R S_{ij}^\theta + a_{ij}^\perp \tag{1.16}$$

also achieves a well-conditioned solution. We follow the nomenclature of Duraisamy *et al.* (2019) in that the θ superscript indicates a quantity that comes from a high-fidelity source such as DNS. The R superscript indicates a quantity taken from the corresponding baseline RANS simulation.

The decomposition in (1.16) permits an augmented turbulence closure framework that treats the machine learning correction only in an explicit term in the momentum equation. Separating the machine learning model prediction has several advantages. It allows the model correction to be easily 'turned off' in unstable situations. It also is more interpretable – rather than correcting both the eddy viscosity and also injecting an explicit correction term, the correction is contained entirely within an explicit term in the momentum equation. Lastly, it avoids the necessity for an optimal eddy viscosity to be computed from high-fidelity data. Although there are methods to increase the practicality of computing the optimal eddy viscosity (McConkey *et al.* 2022*a*), this quantity is often unstable and difficult to predict via a machine learning model.

Figures 1 and 2 demonstrate a conditioning test similar to the tests conducted by Brener *et al.* (2021). Several different decompositions of the DNS Reynolds stress tensor are injected into a RANS simulation, to identify which decompositions permit a well-conditioned solution. Comparing panels (*b*) and (*c*) in both figures 1 and 2, we can see that the proposed decomposition of the Reynolds stress tensor (1.16) achieves an equally well-conditioned solution as the optimal eddy viscosity framework. To our knowledge, this is the first result in the literature showing that a well-conditioned solution can be achieved without the use of an optimal eddy viscosity to incorporate information about the DNS velocity field (Brener *et al.* 2021). We further confirm the findings of Brener *et al.* (2021) with respect to the requirement that the closure decomposition includes information about the DNS mean velocity field in order to achieve a well-conditioned solution. We confirm the notion from Wu *et al.* (2019*a*) that implicit treatment helps address the ill-conditioning issue, but using the DNS strain rate tensor S_{ij}^θ to calculate the explicitly injected $a_{ij}^\perp$.

1.3. *Realisability-informed training*

The Reynolds stress tensor is symmetric positive semidefinite. A set of constraints on the non-dimensional anisotropy tensor b_{ij} arise from this property, as determined by Banerjee *et al.* (2007). These constraints are

$$-\frac{1}{3} \leqslant b_{ij} \leqslant \frac{2}{3}, \quad i = j, \tag{1.17}$$

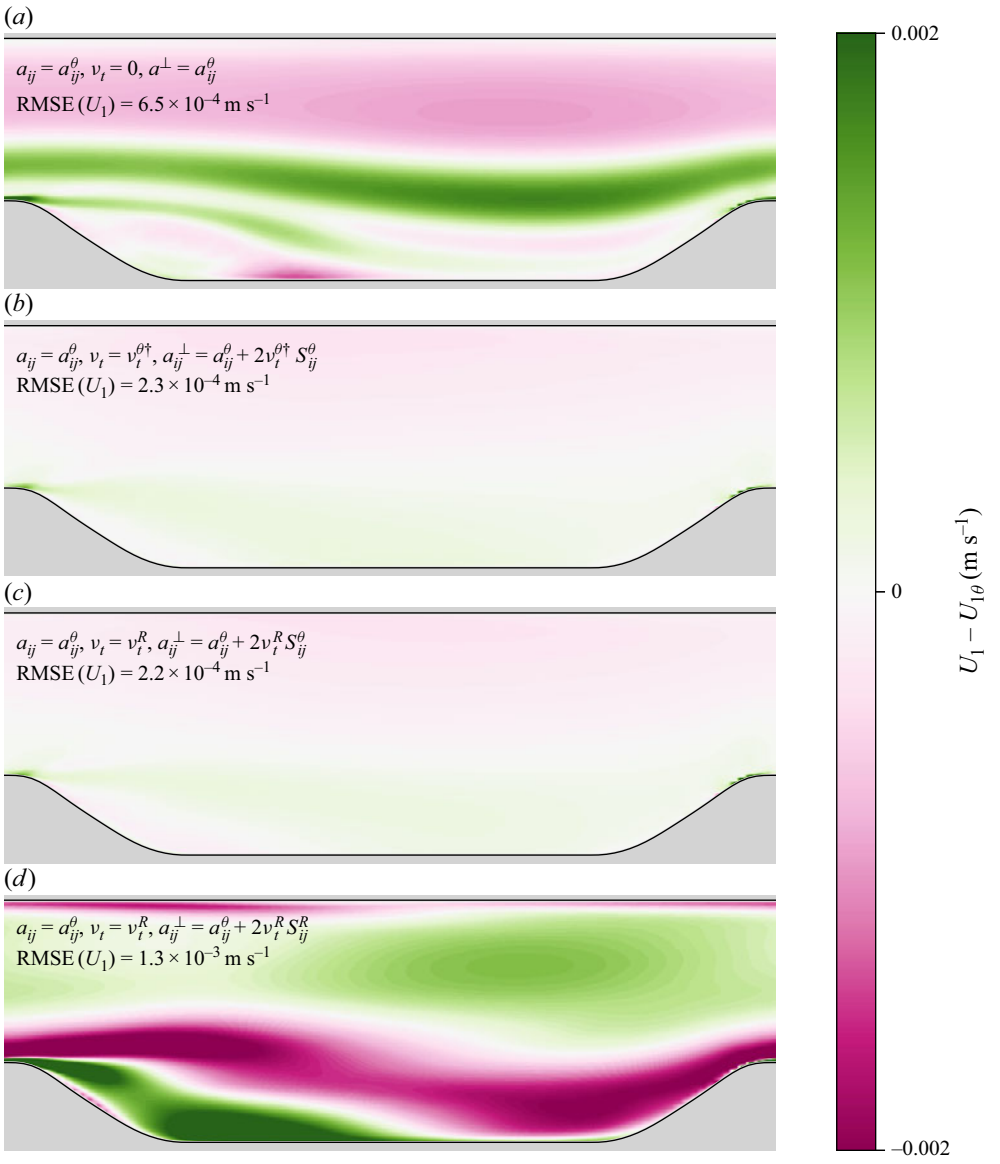


Figure 1. Conditioning test comparing the errors in U_1 after injecting: (*a*) the DNS anisotropy tensor fully explicitly, (*b*) the optimal eddy viscosity (implicitly) and the remaining nonlinear part of the anisotropy tensor calculated using S^{θ}_{ij} (explicitly), (*c*) the eddy viscosity estimated by the k-ω shear stress transport (SST) model (implicitly) and the remaining nonlinear part of the anisotropy tensor calculated using S^{θ}_{ij} (explicitly) and (*d*) the eddy viscosity estimated by the k-ω SST model (implicitly), and the remaining nonlinear part of the anisotropy tensor calculated using S^{R}_{ij} (explicitly), k is the turbulent kinetic energy (TKE), ω is the TKE specific dissipation rate and RSME is the Root mean squared error.

$$-\frac{1}{2} \leqslant b_{ij} \leqslant \frac{1}{2}, \quad i \neq j, \tag{1.18}$$

$$\lambda_1 \geqslant \frac{3|\lambda_2| - \lambda_2}{2}, \tag{1.19}$$

$$\lambda_1 \leqslant \frac{1}{3} - \lambda_2, \tag{1.20}$$

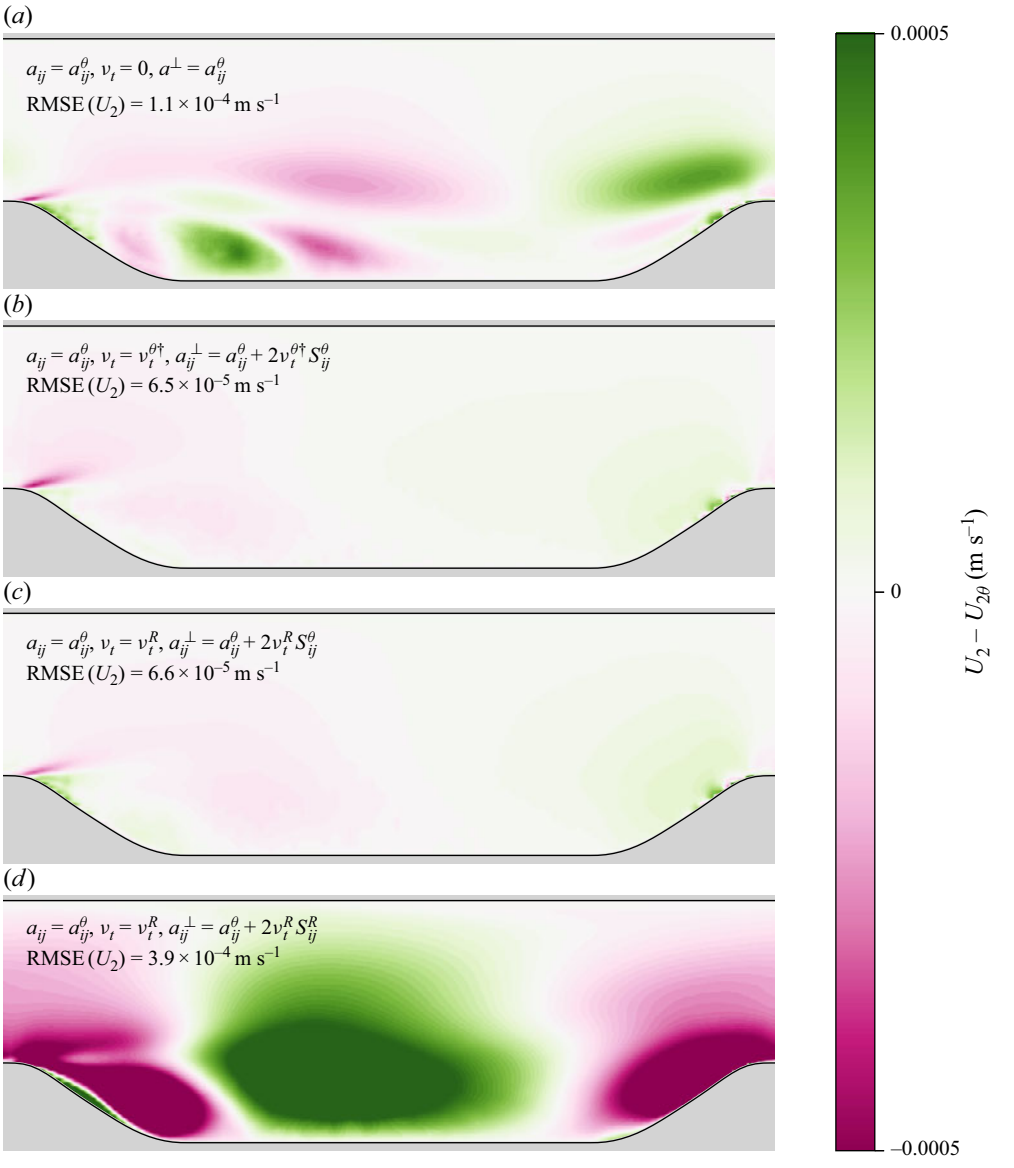


Figure 2. Conditioning test comparing the errors in U_2 after injecting: (*a*) the DNS anisotropy tensor fully explicitly, (*b*) the optimal eddy viscosity (implicitly) and the remaining nonlinear part of the anisotropy tensor calculated using S^{θ}_{ij} (explicitly), (*c*) the eddy viscosity estimated by the k-ω SST model (implicitly) and the remaining nonlinear part of the anisotropy tensor calculated using S^{θ}_{ij} (explicitly) and (*d*) the eddy viscosity estimated by the k-ω SST model (implicitly), and the remaining nonlinear part of the anisotropy tensor calculated using S^{R}_{ij} (explicitly).

where the non-dimensional anisotropy tensor b_{ij} is calculated by

$$b_{ij} = \frac{a_{ij}}{2k}, \tag{1.21}$$

and the eigenvalues of b_{ij} are given by $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3$.

A given Reynolds stress tensor is physically realisable if it satisfies these constraints. While the physical realisability of the closure term may seem an important constraint

for turbulence models, many commonly used turbulence models such as the k-ε model, ε is the TKE dissipation rate (Launder & Spalding 1974), k-ω (Wilcox 1988) and k-ω SST model (Menter 1994; Menter, Kuntz & Langtry 2003) do not guarantee physical realisability.

Based on the widespread acceptance and popularity of non-realisable turbulence models, it is fair to say that realisability is not a hard constraint on a new turbulence model. Nevertheless, the true Reynolds stress tensor is realisable, and if a machine learning model is able to learn to predict realisable closure terms, it may be more physically accurate. Unfortunately, Pope's tensor basis expansion for the anisotropy tensor (and machine learning architectures based on this expansion) do not provide a means to achieve realisability. To enforce realisability, a variety of *ad hoc* strategies have been used, including in the original TBNN paper by Ling *et al.* (2016). Most of these strategies involve postprocessing predictions by the TBNN, such as shrinking the predicted anisotropy tensor in certain directions until it is physically realisable (Jiang *et al.* 2021).

In the present work, we propose including a penalty for violating realisability constraints in the loss function. In a similar spirit of physics-informed neural networks (PINNs) (Raissi, Perdikaris & Karniadakis 2019), we term the use of this loss function 'realisability-informed training'. Whereas PINNs encourage the model to learn physics by penalising violations of conservation laws, realisability-informed TBNNs learn to predict physically realisable anisotropy tensors.

The realisability penalty $\mathcal{R}(b_{ij})$ is given as follows:

$$\begin{aligned}\mathcal{R}(\tilde{b}_{ij}) = \frac{1}{6}\Bigg\{&\sum_{i=j}\left(\max\left[\tilde{b}_{ij}-\frac{2}{3}, -\left(\tilde{b}_{ij}+\frac{1}{3}\right), 0\right]\right)^2 \\ &+ \sum_{i\neq j}\left(\max\left[\tilde{b}_{ij}-\frac{1}{2}, -\left(\tilde{b}_{ij}+\frac{1}{2}\right), 0\right]\right)^2\Bigg\} \\ &+ \frac{1}{2}\Bigg\{\left(\max\left[\frac{3|\lambda_2|-|\lambda_2|}{2}-\lambda_1, 0\right]\right)^2 \\ &+ \left(\max\left[\lambda_1-\left(\frac{1}{3}-\lambda_2\right), 0\right]\right)^2\Bigg\}, \end{aligned} \tag{1.22}$$

where the $\sim$ above a symbol denotes a model prediction for the quantity associated with the symbol.

Equation (1.22) can be thought of as the mean squared violation in the components of $\tilde{b}_{ij}$, plus the mean squared violation in the eigenvalues of $\tilde{b}_{ij}$. To help visualise this penalty function, figure 3 shows the penalties incurred by violating various realisability constraints on the anisotropy tensor.

It should be noted that in the same way that a PINN's prediction cannot be guaranteed to satisfy a conservation law, a realisability-informed TBNN cannot be guaranteed to predict a physically realisable anisotropy tensor. The goal is that the incorporation of a physics-based loss at training time will encode into the model a tendency to predict physically realisable anisotropy tensors. At training time, when the model predicts a b_{ij} component outside the realisability zone, then it is penalised in two ways: the error in this prediction will be non-zero (since all label data are realisable), and the realisability penalty will be

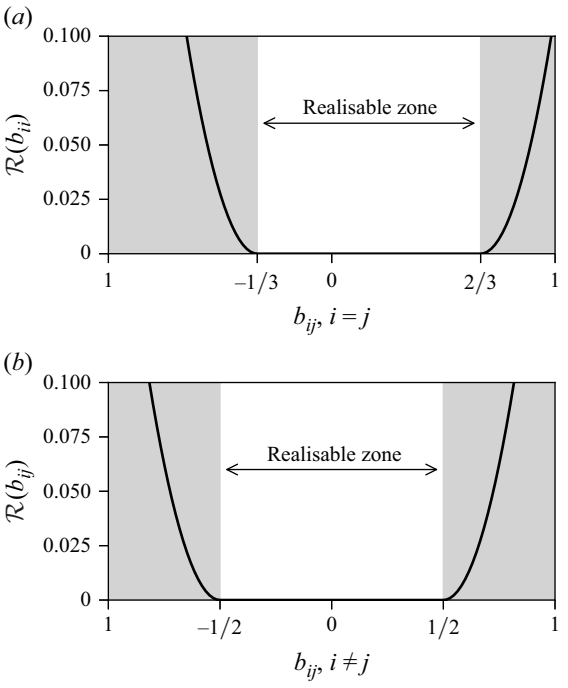


Figure 3. Realisability penalty for the (*a*) diagonal components of b_{ij}, and (*b*) off-diagonal components of b_{ij}.

non-zero. Therefore, the realisability penalty serves as an additional driving factor towards the realisable, true b_{ij} value. The merits of this approach are demonstrated in § 2.2.

Another important metric in the loss function is the error-based penalty $\mathcal{E}(\tilde{b}_{ij})$. When the model predicts a certain anisotropy tensor, it is evaluated against a known high-fidelity anisotropy tensor available in the training dataset. Typically, mean-squared error loss functions are used to train machine learning-augmented closure models. However, we propose the following modifications to the loss function of a TBNN.

(i) Since b_{ij} is a symmetric tensor, we propose to sum the squared errors as follows:

$$\mathcal{E}(\tilde{b}_{ij}) = \frac{1}{6}\left\{\sum_{ij\in\{11,12,13,\,22,23,33\}} \left(\tilde{b}_{ij} - b_{ij}^{\theta}\right)^2\right\}. \tag{1.23}$$

Calculating the squared error in this way avoids double penalising the off-diagonal components, a situation which arises when summing over all components of b_{ij}.

(ii) Although the TBNN model predicts b_{ij}, we propose to use a loss function based on the error in a_{ij}. Near the wall, $\hat{T}_{ij}^{(n)} \to 0$, but $b_{ij} \nrightarrow 0$. The vanishing of $\hat{T}_{ij}^{(n)}$ with non-vanishing b_{ij} causes instabilities during training, leading to $g_n \to \infty$ here. Since a_{ij} is the tensor injected into the momentum equation, its accurate prediction should be the focus of the training process. To dimensionalise b_{ij}, the turbulent kinetic energy k must be used. This leads to the following error-based loss:

$$\mathcal{E}(\tilde{a}_{ij}) = \frac{1}{6}\left\{\sum_{\substack{ij\in\{11,12,13,\\22,23,33\}}} \left(2k^{\theta}\tilde{b}_{ij} - 2k^{\theta}b_{ij}^{\theta}\right)^2\right\} = \frac{2(k^{\theta})^2}{3}\left\{\sum_{\substack{ij\in\{11,12,13,\\22,23,33\}}} \left(\tilde{b}_{ij} - b_{ij}^{\theta}\right)^2\right\}, \tag{1.24}$$

or stated more simply

$$\mathcal{E}(\tilde{a}_{ij}) = (2k^{\theta})^2\mathcal{E}(\tilde{b}_{ij}). \tag{1.25}$$

The reason for the use of k^{θ} in (1.24) and (1.25) is discussed in § 1.4.

The final loss function includes an error-based metric and a realisability-violation penalty. This loss function $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{N}\sum_{p}\frac{\left(2k_p^{\theta}\right)^2}{\sum_{k=1}^{s} Z_k^2\mathcal{I}_{C_k}(p)}\left(\mathcal{E}\left(\tilde{b}_{ij,p}\right) + \alpha\mathcal{R}\left(\tilde{b}_{ij,p}\right)\right), \tag{1.26}$$

where α is a factor used to control the relative importance of the realisability penalty. Here, N is the total number of points in the dataset. The points are indexed by $p = 1, 2, \ldots, N$. The cases in the dataset are indexed $k = 1, 2, \ldots, s$. $\mathcal{I}_{C_k}(p)$ is an indicator function

$$\mathcal{I}_{C_k}(p) = \begin{cases} 1, & \text{if} p \in C_k, \\ 0 & \text{otherwise.} \end{cases} \tag{1.27}$$

For a given point p, $\mathcal{I}_{C_k}(p)$ selects all points which come from the same case as p. The denominator for all points from the same case C_k is the same – normalisation is applied on a case-by-case basis. A given point is normalised by the mean-squared Frobenius norm of the anisotropy tensor over all the points from the case it comes from. The mean Frobenius norm over a case is given by

$$Z_k = \frac{1}{|C_k|}\sum_{p\in C_k}\|a_{ij}(p)\|_F\,, \qquad k = 1, 2, \ldots, s\,, \tag{1.28}$$

where $|C_k|$ is the cardinality of the case C_k (*viz.*, the number of points in the kth case C_k). Here, Z_k is simply the average of the Frobenius norm $\|\cdot\|_F$ of the anisotropy tensor a_{ij} for all points p (*viz.*, $a_{ij}(p)$) in case C_k. The objective of this denominator in (1.26) is to promote a more balanced regression problem, since data points from various cases or flow types may have $||a_{ij}||$ that differ by orders of magnitude. Normalisation on a case-by-case basis is made in an effort to normalise all a_{ij} error magnitudes to a similar scale.

In this study, we use $\alpha = 10^2$ to encode a high preference for realisable results in § 2. Lower values of α will reduce the penalty applied to realisability violations, which may be necessary for flows in which the anisotropy tensor is difficult to predict via a TBNN. As discussed, the multiplicative term $(2k^{\theta})^2$ is used to formulate the loss function in terms of predicting a_{ij} rather than b_{ij}. However, $\mathcal{R}(\tilde{b}_{ij})$ is also multiplied by $(2k^{\theta})^2$ to ensure that the realisability penalty and mean-squared error in a_{ij} are of similar scales.

1.4. *Neural network architecture*

Motivated by improving the training and injection stability, as well as generalisability, we propose several modifications to the original TBNN (Ling *et al.* 2016).

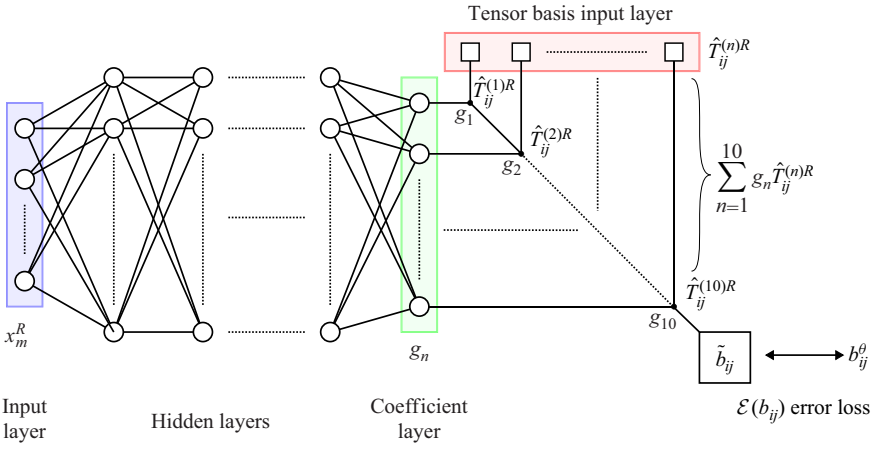


Figure 4. Model architecture and training configuration for the original TBNN proposed by Ling *et al.* (2016).

The original TBNN is shown in figure 4. At training time, this network predicts the non-dimensional anisotropy tensor b_{ij}. All basis tensors used in this prediction at training time come from RANS, and during training the prediction $\tilde{b}_{ij}$ is evaluated against a known value of b_{ij}^{θ} from a high-fidelity simulation. At injection time, the network is used in this same configuration to predict $\tilde{b}_{ij}$. Here, $\tilde{b}_{ij}$ is injected into a coupled system of equations consisting of the continuity/momentum equations (explicit injection), as well as the turbulence transport equations. This system of equations is iterated around a fixed $\tilde{b}_{ij}$, to obtain an updated estimate for the turbulent kinetic energy k, and therefore an updated estimate for $\tilde{a}_{ij} = 2k\tilde{b}_{ij}$.

The modified TBNN is shown in figure 5. This TBNN relies on the same tensor basis expansion as the original TBNN. However, the linear term has been modified in this expansion. Whereas the original TBNN uses

$$\hat{T}_{ij}^{(1)} = \frac{k^R}{\varepsilon^R} S_{ij}^R, \tag{1.29}$$

our modified TBNN uses

$$\hat{T}_{ij}^{(1)} = \frac{\nu_t^R}{k^{\theta}} S_{ij}^{\theta}. \tag{1.30}$$

Further, while the original TBNN calculates the linear (denoted by superscript L) component of b_{ij} as

$$b_{ij}^L = g_1 \hat{T}_{ij}^{(1)} = g_1 \frac{k}{\varepsilon} S_{ij}, \tag{1.31}$$

our modified TBNN uses

$$b_{ij}^L = -\hat{T}_{ij}^{(1)} = -\frac{\nu_t}{k} S_{ij}, \tag{1.32}$$

where the superscript R denotes a quantity that comes from the original RANS simulation.

These changes are motivated by the following.

(i) At injection time, we use implicit treatment of the linear term $\hat{T}_{ij}^{(1)}$ to formulate (1.15) in a stable manner. After injection, S_{ij} will continue to evolve. In a similar spirit

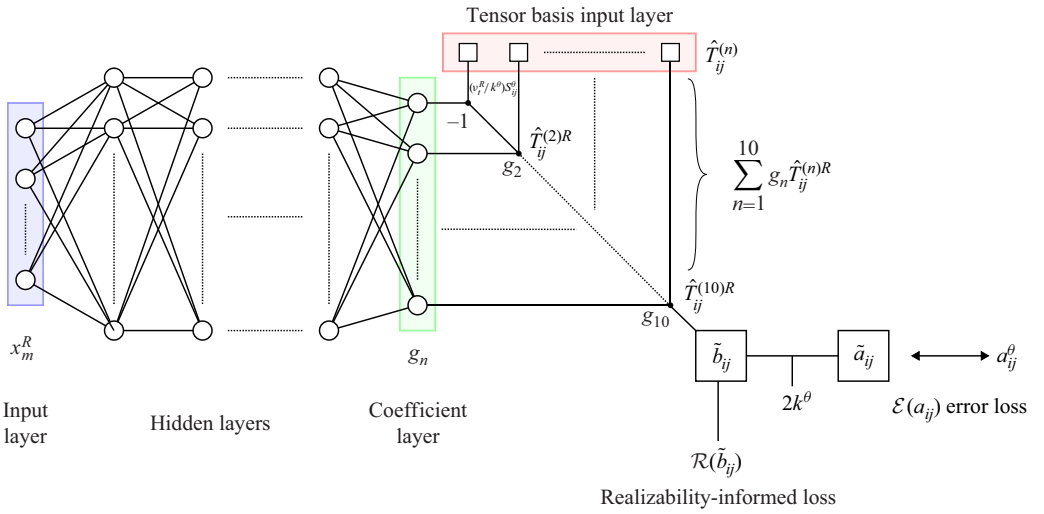


Figure 5. Model architecture and training configuration proposed in the present investigation. Note that during training, S_{ij}^{θ} is used in $\hat{T}_{ij}^{(1)}$, whereas all other quantities come from the original RANS simulation.

as optimal eddy viscosity frameworks, we therefore use S_{ij}^{θ} to compute the linear component at training time. In optimal eddy viscosity based frameworks, using $S_{ij} = S_{ij}^{\theta}$ at training time helps drive $U_j \to U_j^{\theta}$ at injection time (the cause of this behaviour is currently unknown). In Ling *et al.* (2016), all basis tensors (and therefore b_{ij}) remain fixed after injection. We also fixed $\nu_t = \nu_t^R$ at injection time (*viz.*, no further evolution of the eddy viscosity is permitted).

(ii) At training time, we dimensionalise $\tilde{b}_{ij}$ using k^{θ}: $\tilde{a}_{ij} = 2k^{\theta}\tilde{b}_{ij}$. At evaluation time, we do not have k^{θ}. However, the need for k^{θ} is avoided, since we only use the nonlinear part of $\tilde{b}_{ij}$ at test time. The reason we only need $\tilde{b}_{ij}^{\perp}$ at test time is that the training process has been designed to use the linear part of b_{ij} estimated by the RANS turbulence model, and augment this by the TBNN's equivariant prediction for $\tilde{b}_{ij}^{\perp}$.

(iii) Using ν_t/k to normalise the basis tensors and fixing $g_1 = -1$ in (1.32) results in the RANS prediction for b_{ij}^L being implicitly used in the TBNN. Therefore, the TBNN learns to correct b_{ij} using $b_{ij}^{\perp}$ in a way that allows a realisability-informed training process, and fully implicit treatment of the linear term at injection time.

Lastly, the use of k^{θ} to dimensionalise $\tilde{b}_{ij}$ is also enabled by our use of a separate neural network to correct k^R at injection time. This neural network is called the k-correcting neural network (KCNN), and is a simple fully connected feed-forward neural network that predicts a single output scalar Δ (Figure 6)

$$\Delta = \log\left(\frac{k^{\theta}}{k^R}\right), \tag{1.33}$$

such that at injection time, an updated estimate for k can be obtained $\tilde{k} = e^{\Delta}k^R$, without the need to re-couple the turbulence transport equations. The KCNN shares the same input features as the TBNN.

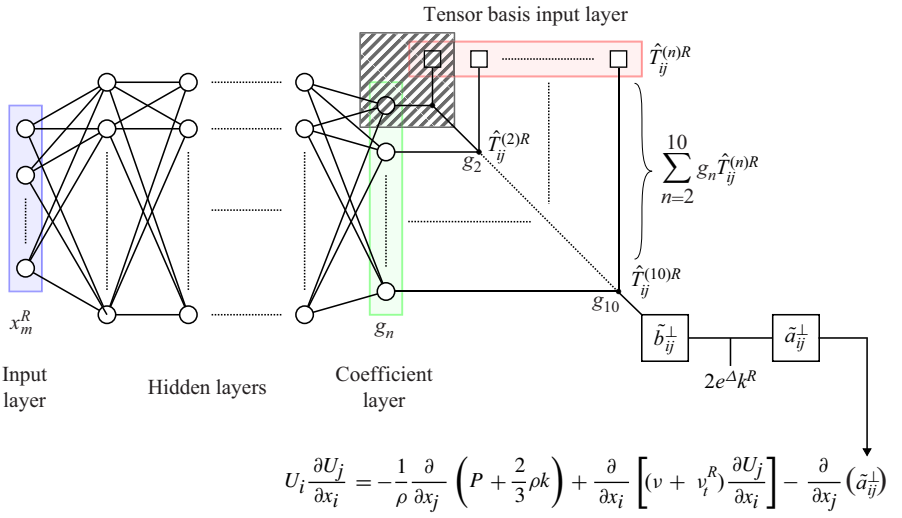


Figure 6. Test-time injection configuration proposed in the present investigation. The hatching over g_1 and $\hat{T}_{ij}^{(1)}$ indicates that they are not used at injection time.

Together, the KCNN and TBNN predict the anisotropy tensor in the following manner:

$$\tilde{a}_{ij} = 2\left(e^{\Delta}k^R\right)\sum_{n=1}^{10} g_n \hat{T}_{ij}^{(n)}. \tag{1.34}$$

With the linear component of $\tilde{a}_{ij}$ being treated implicitly during injection, the entire closure framework is summarised as explicit injection of the following term into the momentum equation:

$$\tilde{a}_{ij}^{\perp} = 2\left(e^{\Delta}k^R\right)\sum_{n=2}^{10} g_n \hat{T}_{ij}^{(n)R}. \tag{1.35}$$

1.4.1. *Tensor basis Kolmogorov–Arnold network (TBKAN)*

The tensor basis KAN shown in figure 7 in the training configuration replaces the multi-layer perceptron in the modified TBNN with a KAN introduced by Liu *et al.* (2024). The Kolmogorov–Arnold representation theorem states that any continuous multivariate function can be expressed as a composition of continuous univariate functions. KANs are based on this theorem, replacing the typical linear weight matrices in neural networks with learnable one-dimensional (1-D) functions (Liu *et al.* 2024). These functions are parameterised using splines, offering a flexible and computationally efficient approach to represent continuous functions. The KANs utilise B-splines, which are piecewise polynomial functions, to model local variations in data. Each spline segment corresponds to a polynomial function, and their piecewise nature allows KANs to approximate the local variations in the data during training. This adaptability enables KANs to capture intricate functional relationships more effectively, merging the advantages of B-splines with the traditional neural network framework, thereby enhancing both accuracy and interpretability.

In the TBKAN architecture, KAN replaces the hidden layers of the standard TBNN, while the anisotropy mapping portion remains unchanged. The TBKAN's output layer is designed to predict the coefficients of the tensor basis.

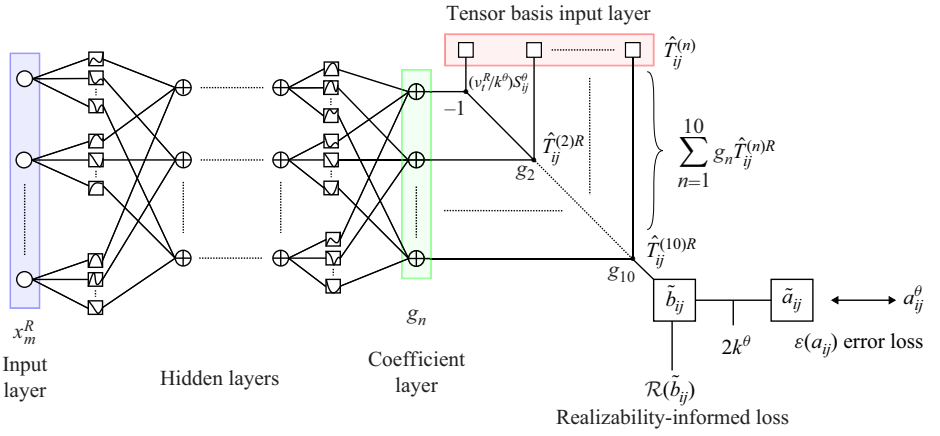


Figure 7. Model architecture and training configuration for TBKAN which replaces the multi-layer perceptron in the modified TBNN (see figure 5) with a KAN.

1.5. *Machine learning procedure*

The KCNN, TBNN and TBKAN architectures proposed in § 1.4 were implemented in PyTorch, along with the proposed realisability-informed loss function (1.26). These models were trained using an open-source dataset for data-driven turbulence modelling (McConkey, Yee & Lien 2021). The objective of this study is to train and evaluate models trained on various flows, to determine whether the proposed realisability-informed loss function and architecture modifications are significantly beneficial. All code is available on Github (McConkey & Kalia 2024).

1.5.1. *Datasets*

The training flows consist of flow over periodic hills (Xiao *et al.* 2020), flow through a square duct (Pinelli *et al.* 2010) and flow over a flat plate with zero pressure gradient (Rumsey 2021). These flows are selected because they contain several challenging physical phenomena for RANS, including separation, reattachment and Prandtl's secondary flows. The flat plate case is also included, to demonstrate how machine learning can improve the anisotropy estimates within the boundary layer. For each flow type, both a hold-out validation set and a hold-out test set are selected. The validation set is used during training to help guide when to stop training in order to prevent overfitting, but the validation set loss is not back propagated through the network to update weights and biases. While we hold out an entire case for the test set (the usual procedure in data-driven turbulence modelling), we also generate the validation sets by holding out entire cases at a time. We recommend this method for generating validation sets in data-driven turbulence modelling – it is analogous to grouped cross-validation, a practice used in machine learning where several data points come from a single observation. Here, we consider each separate flow case as a single observation, each containing many data points. It is therefore prudent to ensure that two data points from the same observation are not used in both the training and test set.

Table 1 outlines the three training/validation/test splits considered. The objective in splitting the dataset this way is to determine whether a realisability-informed model can generalise to a new case for a given flow. Machine learning-based anisotropy mappings do not generalise well to entirely new flows (Duraisamy 2021; McConkey *et al.* 2022*b*; Man *et al.* 2023). However, they can be used to dramatically enhance the performance of a RANS simulation for a given flow type. In this same spirit, we aim to test how our

	Flat plate	Square duct	Periodic hills
Data points	1 396	1 47 456	73 755
Parameter varied	Re_θ	Re_H	α
Training set	1000, 2000, 2540, 3270, 3970	1100, 1150, 1250, 1350, 1400, 1500, 1600, 2205, 2400, 2600, 2900, 3500	0.5, 1.0, 1.5
Validation set	1410, 3030, 4060	1300, 1800, 3200	0.8
Test set	3630	2000	1.2

Table 1. Datasets used for training, validation and testing. Varied parameters are defined in § 2.1.

modifications improve the generalisability of the learned anisotropy mapping to an unseen flow, albeit within the same class of flow.

1.5.2. *Input features*

The input features here are all derived from the baseline RANS k-ω SST simulation. The input features form the vector x_m^R. The superscript R has been dropped in this section to avoid crowded notation, but it applies to all quantities discussed in § 1.5.2.

The input features must be Galilean invariant in order to generate an appropriately constrained anisotropy mapping. Most data-driven anisotropy mapping investigations use a mixture of heuristic scalars and scalars systematically generated from a minimal integrity basis for a set of gradient tensors (Wu *et al.* 2018). We emphasise that all scalars must be Galilean invariant – without this criterion, the RANS equations will lose Galilean invariance. Despite the importance of this constraint, several data-driven anisotropy mappings include scalars like the turbulence intensity, which breaks Galilean invariance.

We use a mixture of heuristic scalars and scalars systematically generated from a minimal integrity basis (Wu *et al.* 2018). We use the following heuristic scalars:

$$q_1 = \min\left(\frac{\sqrt{k}y_w}{50\nu}, 2\right), \tag{1.36}$$

$$q_2 = \frac{k}{\varepsilon}\sqrt{\sum_i \sum_j |S_{ij}|^2}, \tag{1.37}$$

$$q_3 = \frac{\sqrt{\sum_i \sum_j |\tau_{ij}|^2}}{k}, \tag{1.38}$$

$$q_4 = \frac{\sqrt{k}}{0.09\omega y_w}, \tag{1.39}$$

$$q_5 = \frac{500\nu}{y_w^2\omega}, \tag{1.40}$$

$$q_6 = \min\left(\max\left(q_4, q_5\right), \frac{2.0k}{\max\left(\frac{y_w^2}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}\right)}\right), \tag{1.41}$$

with $\varepsilon = 0.09\omega$, and y_w is the distance to the nearest wall.

Model	Dataset	Learning rate	Epochs	Hidden layers	Neurons per layer
TBNN	Flat plate	$5(10)^{-4}$	7959	4	20
KCNN	Flat plate	$2.5(10)^{-4}$	2478	5	30
TBNN	Square duct	$5(10)^{-4}$	100	7	30
KCNN	Square duct	$5(10)^{-4}$	1150	7	30
TBNN	Periodic hills	$1(10)^{-5}$	19140	7	30
KCNN	Periodic hills	$1(10)^{-6}$	13138	11	30

Table 2. Hyperparameters selected for each model.

These input features correspond to: the wall-distance-based Reynolds number (q_1), the ratio of turbulent time scale to mean strain time scale (q_2), the ratio of total Reynolds stress to k (q_3) and different blending scalars used within the k-ω SST model (q_4, q_5, q_6). The full list of integrity basis tensors is given in Appendix A. We use the first invariant (the trace, A_{ii}) of the following tensors: $B_{ij}^{(1)}$, $B_{ij}^{(3)}$, $B_{ij}^{(4)}$, $B_{ij}^{(6)}$, $B_{ij}^{(7)}$, $B_{ij}^{(16)}$ and $B_{ij}^{(35)}$. We use the second invariant, $1/2((A_{ii})^2 - A_{ij}A_{ji})$, of the following tensors: $B_{ij}^{(3)}$, $B_{ij}^{(6)}$, $B_{ij}^{(7)}$, $B_{ij}^{(8)}$. These input features were hand picked from the full set of 94 invariants listed in Appendix A (Wu *et al.* 2018; McConkey *et al.* 2022*a*). As discussed in McConkey *et al.* (2022*a*), many of the invariants are zero for 2-D flows. However, different conditions cause different invariants to be zero. For example, in the present study, there are a different set of zero invariants for flow through over periodic hills, and flow through a square duct. This difference occurs because different components of $\partial()/\partial x_j$ are to be zero. We have performed a systematic investigation using a symbolic math toolbox (sympy (Meurer *et al.* 2017)) to determine which invariants are zero for the duct case, and general 2-D flows. We make the results and code available in Appendix A and on Github (McConkey 2023), respectively. Input features used in this investigation were selected based on the results in Appendix A. Therefore, the input features are not uniformly zero on any of the considered flows.

To ensure all input features are of the same magnitude during training, they are scaled according to the following formula:

$$x_m^R = \frac{X_m^R - \mu_m}{\sigma_m}, \tag{1.42}$$

where x_m^R is the input feature vector for the neural network, X_m^R is the raw input feature vector from the RANS simulation, μ_m is a vector containing the mean of each input feature over the entire training dataset and σ_m is a vector containing the standard deviation of each input feature over the entire training dataset.

When making predictions on the hold-out validation and test sets, the mean and standard deviation values from the training data are used to avoid data leakage.

1.5.3. *Hyperparameters and training procedure*

The hyperparameters for each neural network were hand tuned based on validation set performance. The hidden layers for all TBNNs and KCNNs are fully connected, feed-forward layers with Swish activation functions (Ramachandran, Zoph & Le 2017). The appropriate hyperparameters vary between flows, since each dataset contains a different number of data points, and the anisotropy mapping being learned is distinct. Table 2 shows the final hyperparameters used. All training runs used a mini-batch size of 32, except the periodic hill TBNN run, which used a mini-batch size of 128.

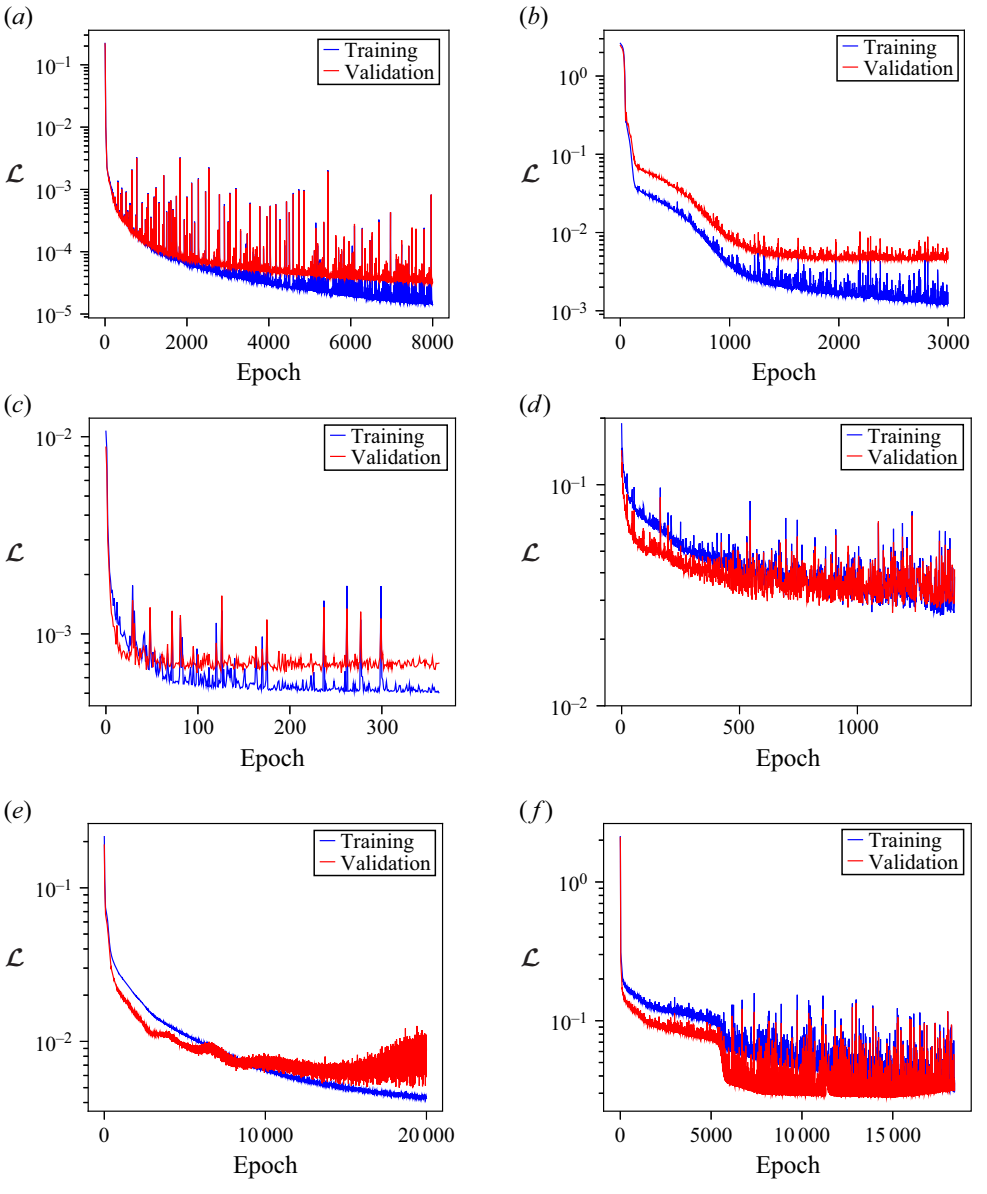


Figure 8. Loss function vs epoch for the (*a*) TBNN and (*b*) KCNN models on the flat plate dataset; the (*c*) TBNN and (*d*) KCNN models on the square duct dataset; and the (*e*) TBNN and (*f*) KCNN models on the periodic hill dataset.

The AMSGrad Adam optimiser (Reddi, Kale & Kumar 2018) was found to achieve better performance than the standard Adam optimiser (Kingma & Ba 2015) for training TBNNs. The learning rate and number of epochs for each optimiser is given in table 2. Satisfactory performance was achieved with a constant learning rate; for training on more complex flows we recommend the use of learning rate scheduling to achieve better performance. The training/validation loss curves for each model are shown in figure 8.

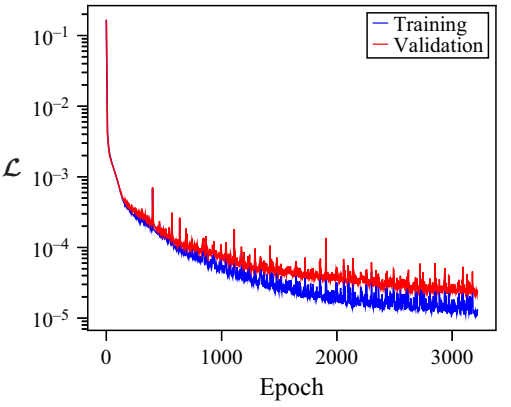


Figure 9. Loss function vs epoch for the TBKAN model trained on the flat plate dataset, with the best-performing KAN configuration given by a network width of $w = 9$, a grid size of $g = 8$ and a polynomial order of $k = 3$ (cubic B-spline).

1.5.4. *Hyperparameter tuning for TBKAN*

The performance of the TBKAN was extensively optimised through a combination of systematic hyperparameter tuning and manual adjustments. The architecture of the TBKAN model was configured as [6, 9, 10], where 6 corresponds to the number of input features, 9 represents the network width (number of neurons per hidden layer) and 10 denotes the output size of the network. The depth of the network, representing the number of hidden layers, was fixed to 2 for the flat plate case.

Hyperparameter tuning focused on refining the grid size (g), network width (w), spline order and input feature combinations. An initial set of 27 runs, conducted with randomly selected configurations, broadly explored the hyperparameter space. These runs were used as a warm start for Bayesian optimisation, which further utilised 143 trials to systematically refine the hyperparameters. The configuration employed a grid size of 8 control points for the B-spline basis representation, with the polynomial order k of the splines fixed at a value of three (*viz.*, $k = 3$ corresponding to cubic splines). This choice was found to provide the best balance between flexibility and computational efficiency.

The final hyperparameters for the best-performing model were as follows: architecture [6, 9, 10], a depth of 2 layers, a grid size of 8 control points and a learning rate of 4.9×10^{-3}. This configuration achieved a mean-squared error (MSE) of 0.22 on the flat plate case. Input feature selection was refined by fixing three core features while sampling from a broader set to enhance the model's adaptability in predicting finer details of the anisotropy tensor. The AMSGrad Adam optimiser (Reddi *et al.* 2018) was used for all training runs, with a mini-batch size of 32, which ensured stable convergence. Training and validation loss curves for the best-performing model for the flat plate, corresponding to a KAN configuration with $w = 9$, $g = 8$ and $k = 3$ (cubic splines), are shown in figure 9.

1.6. *Computational costs*

The inference-time cost for the proposed methodology varies significantly from case to case. A prediction requires (i) running a baseline RANS simulation, (ii) evaluating the Machine learning (ML) model predictions and (iii) running a corrected RANS simulation. Compared with steps (i) and (iii), the cost of step (ii) (model inference) is negligible. Additionally, since the converged fields from step (i) are used to initialise the simulation in step (iii), the cost of the corrected RANS simulation is also reduced. Generally, we found that combined inference-time costs for steps (i), (ii) and (iii) were between 1.5 and

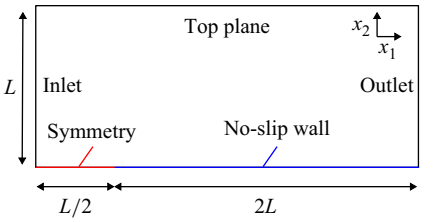


Figure 10. Computational domain for the zero pressure gradient flat plate boundary layer case.

3 times the cost of the baseline simulation (i.e. steps (i), (ii) and (iii) cost approximately 1.5–5 times as much as step (i)), depending on how much the injected closure fields differ from the original linear eddy viscosity-based field. The training cost also varies depending on the dataset. For the periodic hills and square duct datasets here, the training time was approximately 16 GPU hours on a single NVIDIA RTX 3090 GPU.

2. Results for proposed TBNN and TBKAN architectures

2.1. *Generalisation tests*

It was of interest to determine how well the trained models generalise to unseen variations of their training flows. This section demonstrates generalisation results for flow over a flat plate with zero pressure gradient, flow through a square duct and flow over periodic hills.

For all cases, the original RANS solution was generated using OpenFOAM v2212, assuming an isothermal, incompressible and Newtonian fluid. Simulation parameters such as solver, schemes and solution methodology for the zero pressure gradient flat plate case were identical to those discussed in McConkey *et al.* (2021).

It should be noted that the a_{ij} predictions shown for the TBNN/KCNN use S_{ij}^{θ} for predicting the linear part of the anisotropy tensor. The nature of the proposed TBNN training process is to utilise S_{ij}^{θ} during training, so that the remaining nonlinear part can be extracted during injection.

2.1.1. *Flat plate (TBNN)*

This case features a developing turbulent boundary layer on a flat plate with zero pressure gradient, based on the NASA '2DZP' validation case (Rumsey 2021). Figure 10 shows the domain for the flat plate case. The NASA-provided meshes are sufficient to resolve the viscous sublayer region, with a total number of cells $N \approx 2\,00\,000$. However, this mesh was further refined to increase the number of solution data points available for training and testing. The goal of this case is to learn how the anisotropy tensor evolves in a turbulent boundary layer, therefore substantial mesh refinement was required to generate data points in this region. The total number of cells in the mesh is $N = 4\,673\,130$. The plate-length Reynolds number is $Re_L = 5(10)^6$ to match the NASA reference data. The reference data for this case consist of a series of wall-normal profiles, for various Re_θ, defined as

$$Re_\theta = \frac{U_\infty \theta}{\nu}, \tag{2.1}$$

where the momentum thickness θ is given as

$$\theta = \int_0^\infty \frac{U_1}{U_\infty}\left(1 - \frac{U_1}{U_\infty}\right) \mathrm{d}x_2, \tag{2.2}$$

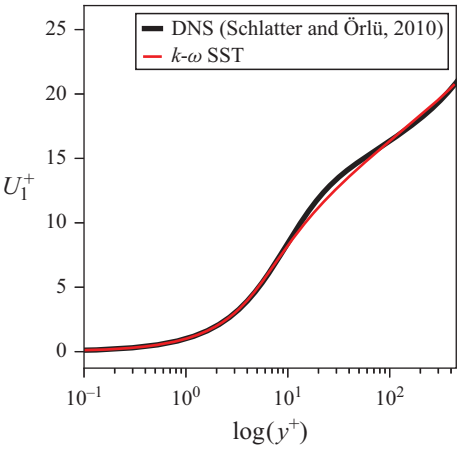


Figure 11. Streamwise velocity profile in the $Re_\theta = 3630$ boundary layer predicted by the k-ω SST model, compared with the reference data from Schlatter & Örlü (2010). Here, $U_1^+ = U_1/u_\tau$, where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity, and τ_w is the wall shear stress. A density of 1 kg m^{-3} was used to be consistent with OpenFOAM's kinematic units. The wall-normal coordinate is $y^+ = x_2 u_\tau/\nu$.

and U_∞ is the free-stream velocity. The following boundary conditions and fluid properties are used for the domain in figure 10. At the inlet boundary, U_j, k and ω are uniform: $U_j = (69.4, 0, 0)$ m s^{-1}, $k = 1.08(10)^{-3}$ m^2 s^{-2}, $\omega = 8675$ s^{-1} and P is zero normal gradient. At the outlet, P is zero, and all other variables are zero normal gradient. At the symmetry plane, all variables are zero normal gradient, and normal velocity is zero. At the top plane, all flow variables are zero normal gradient. At the no-slip wall, $U_j = 0$, $k = 0$, $\omega = 6\nu/\beta_1 y^2$ (here, $\beta_1 = 0.075$) and P is zero normal gradient. A kinematic viscosity of $\nu = 1.388(10)^{-5}$ m s^{-2} was used.

The DNS reference data for a developing turbulent boundary layer come from Schlatter & Örlü (2010). This dataset contains a variety of turbulent boundary layer profiles, at various Re_θ shown in table 1. As discussed in § 1.5.1, the TBNN model was trained on various Re_θ, with $Re_\theta = 3630$ serving as a hold-out test case. The results shown in this section are for this hold-out test case.

The baseline k-ω SST model performs well for the test case flow in terms of predicting the mean velocity profiles. Figure 11 shows a sample mean velocity profile predicted by the baseline k-ω SST model, and figure 12 shows the predicted evolution of Re_θ along the plate. The excellent performance of the k-ω SST model demonstrated by figures 11 and 12 is expected. This case features a fully attached boundary layer with zero pressure gradient, which is one of the fundamental calibration scenarios for RANS models. The zero pressure gradient turbulent boundary layer is considered a 'solved problem' for modern RANS models (Spalart 2023).

While the mean velocity profile is predicted well, figure 13 shows that the evolution of the near-wall anisotropy tensor is not predicted well. For this reason, the model architecture discussed in § 1.4 was used to correct the anisotropy tensor in the near-wall region. This test also aims to determine whether the input feature set is sufficiently expressive to enable predicting the evolution of the anisotropy tensor within a boundary layer. Figure 13 shows the wall-normal profiles of various anisotropy tensor components predicted by the k-ω SST model, the ML-augmented k-ω SST model and the reference DNS simulation for the hold-out test case. As discussed in § 1.5.1, these models were trained on flat plate data at various values of Re_θ. The results in figure 13 are designed to test these models on

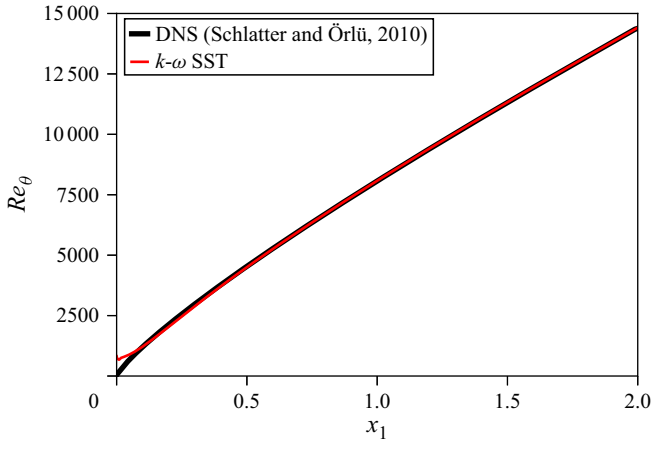


Figure 12. Momentum thickness Reynolds number growth along the flat plate as predicted by the k-ω SST model, and the reference DNS data from Schlatter & Örlü (2010).

input features from an unseen boundary layer profile, to determine whether the learned anisotropy mapping was generalisable.

Figure 13(a) shows the predicted evolution of a_{11} in the turbulent boundary layer. The baseline k-ω SST model predicts $a_{11} = 0$, since $\partial U_1/\partial x_1 \approx 0$ in the boundary layer. However, the DNS data clearly show that a_{11} is non-zero in the boundary layer. The TBNN/KCNN model combination is able to correct the a_{11} term to a high degree of accuracy in the boundary layer on this test case, indicating that the anisotropy mapping for the a_{11} component generalises well. Similar evolutions of a_{22} (figure 13c) and a_{33} (figure 13d) are observed in the DNS data. Again, the k-ω SST model predicts $a_{11} = a_{22} = a_{33} = 0$, which is not physically correct. The TBNN/KCNN models are able to correct the baseline prediction to a high degree of accuracy on this unseen boundary layer profile.

Figure 13(b) shows the predicted evolution of a_{12}. The baseline RANS model predicts the evolution of a_{12} well, and this is likely the reason that the mean velocity profile of U_1 is predicted well (see figure 11). While the TBNN/KCNN is not needed to correct this off-diagonal component, it is able to correct minor inaccuracies in the k-ω SST model predictions in the buffer region ($5 \leqslant y^+ \leqslant 30$). Nevertheless, the baseline k-ω SST model achieves a satisfactory accuracy level for this flow. As discussed, this is expected, given that low Reynolds number RANS models are able to predict a zero pressure gradient boundary layer with a high degree of accuracy. Figure 13 demonstrates that this performance is the result of an accurate prediction of a_{12} by the k-ω SST model.

2.1.2. *Flat plate (TBKAN)*

The predicted evolution of the various components of the anisotropy tensor for the flat plate case obtained using the TBKAN/KCNN model combination is displayed in figure 14(a)–14(d) for the hold-out test case $Re_\theta = 3630$. These predictions are compared with both the baseline k-ω SST model and DNS data from Schlatter & Örlü (2010). For this test case, the predictive accuracy of the TBKAN/KCNN model combination for the anisotropy tensor components is compared with that of the baseline model.

The predicted evolution of a_{11} is shown in figure 14(a). Unlike the k-ω SST model, which predicts $a_{11} = 0$ due to its linear stress assumptions, the TBKAN captures the non-zero nature of a_{11} in the turbulent boundary layer. The TBKAN aligns closely with

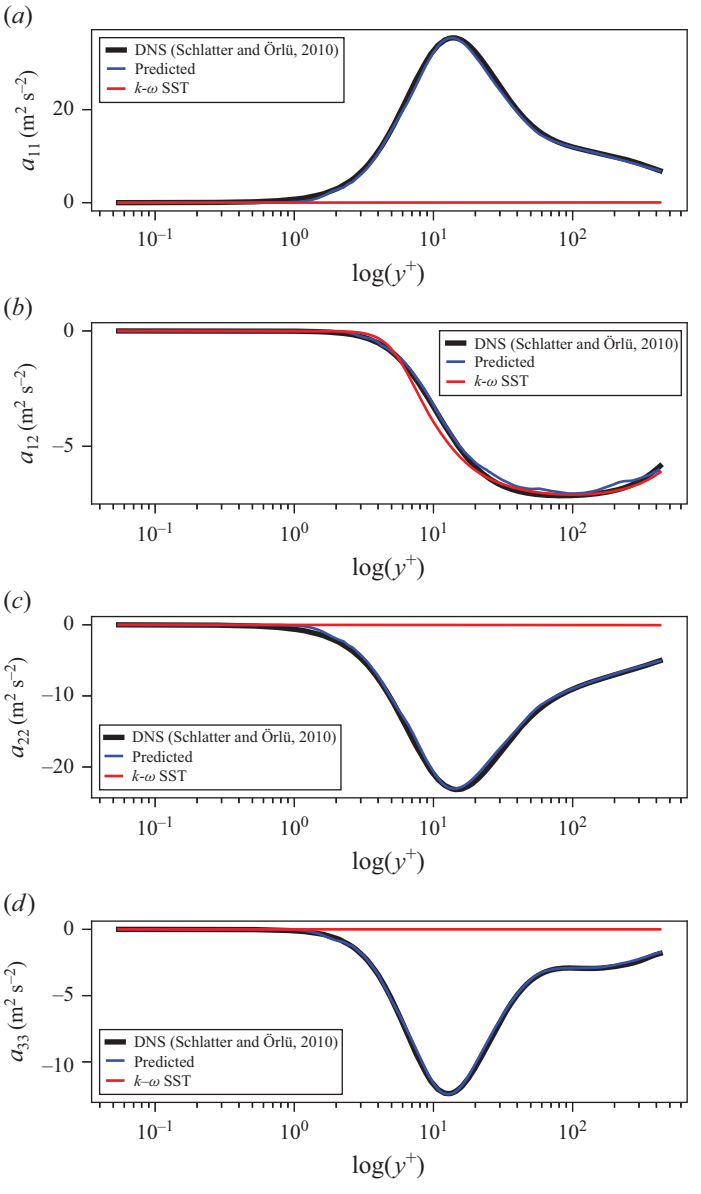


Figure 13. Evolution of (*a*) a_{11}, (*b*) a_{12}, (*c*) a_{22} and (*d*) a_{33} in the $Re_\theta = 3630$ boundary layer, as predicted by the DNS data from Schlatter & Örlü (2010), the TBNN/KCNN model and the baseline k-ω SST model.

the DNS data, indicating its ability to generalise and represent anisotropy accurately, particularly for components where the baseline model is limited.

Figure 14(*b*) presents the predicted evolution of a_{12}. The baseline k-ω SST model predicts this off-diagonal component with reasonable accuracy. However, the TBKAN/KCNN exhibits better conformance with the reference DNS data in the buffer region ($5 \leqslant y^+ \leqslant 30$), correcting minor deviations in the predictions of this quantity provided by the baseline k-ω SST model.

Figures 14(*c*) and 14(*d*) display the predictions of a_{22} and a_{33}, respectively. While the k-ω SST model predicts a value of zero from these two anisotropy stress components,

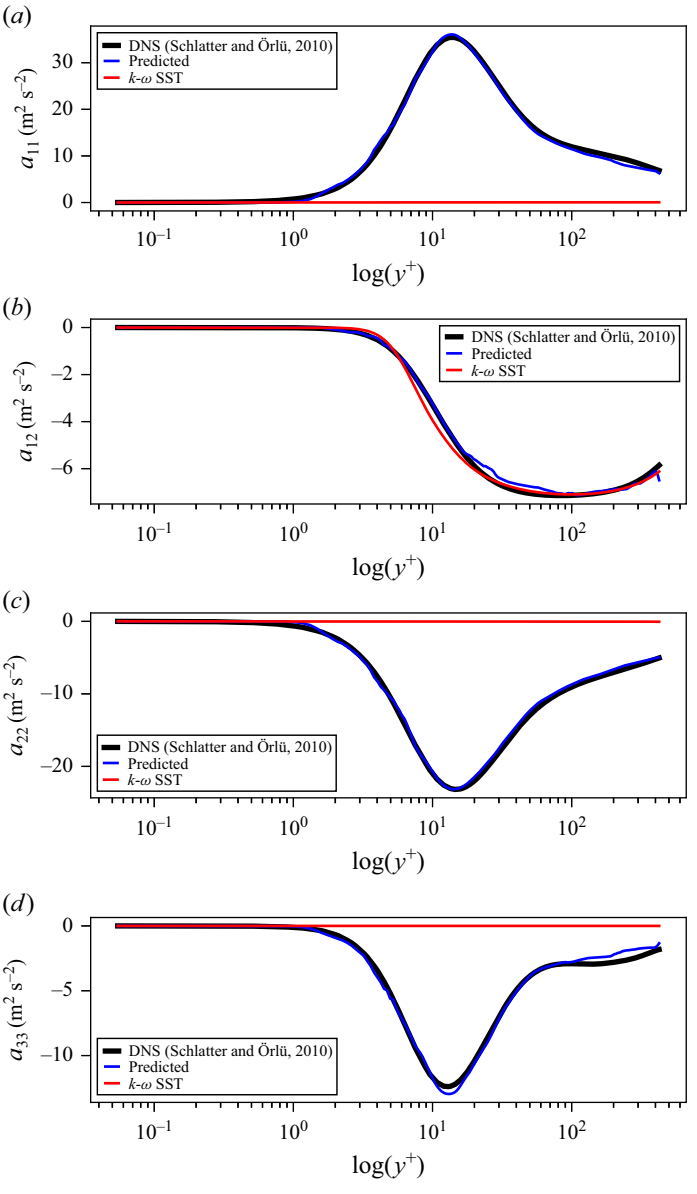


Figure 14. Evolution of (*a*) a_{11}, (*b*) a_{12}, (*c*) a_{22} and (*d*) a_{33} in the $Re_\theta = 3630$ boundary layer, as predicted by the DNS data from Schlatter & Örlü (2010), the TBKAN/KCNN model and the baseline k-ω SST model.

the TBKAN/KCNN captures their evolution with good accuracy compared with the DNS data. Overall, TBKAN/KCNN enhances the predictive accuracy of the anisotropic stress components across the boundary layer. While a_{12} predictions are marginally improved compared with the baseline model, TBKAN/KCNN shows substantial improvements for the normal stress components. Furthermore, a visual perusal of figures 13 and 14 shows that the conformance of the predictions of the anisotropy tensor components with the reference DNS data obtained with TBKAN/KCNN is marginally worse than that obtained with TBNN/KCNN. However, it is noted that the TBKAN is simpler than TBNN in this case in the sense that the former network used only one hidden layer with 9 nodes

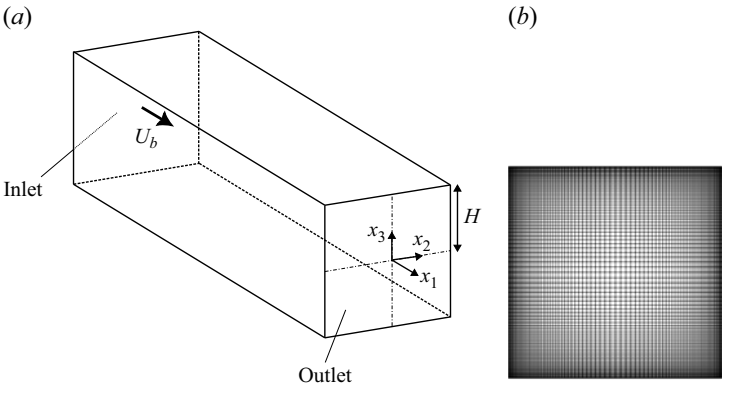


Figure 15. Computational (*a*) domain and (*b*) RANS mesh for the square duct flow.

(where the information from the edges encoded in the B-splines are simply accumulated), whereas the latter network used four hidden layers consisting of 20 nodes each (where the information embodied by the linear weights in the edges is transformed by the nonlinear activation function).

2.1.3. *Square duct*

Turbulent flow through a square duct is a challenging case for RANS models, since linear eddy viscosity models cannot predict the secondary flows that occur in the cross-sectional plane. The goal of the square duct test case is to determine whether the proposed closure framework could enable the k-ω SST model to predict these Prandtl secondary flows (Nikitin, Popelenskaya & Stroh 2021). The square duct DNS dataset generated by Pinelli *et al.* (2010) was used as reference data, and the RANS data from McConkey *et al.* (2021) was used.

Figure 15 shows the computational set-up and mesh for the square duct case. The mesh is designed to achieve $y^+ \leqslant 1$ for all square duct cases. As discussed in § 1.5.1, the duct half-height H Reynolds number varies between cases, calculated by

$$Re_H = \frac{U_b H}{\nu}, \tag{2.3}$$

where U_b is the mean (bulk) cross-sectional velocity. A kinematic viscosity of $\nu = 5(10)^{-6}$ m^2 s^{-1} was used for all square duct cases. With the geometry fixed as shown in figure 15, the bulk velocity was adjusted to vary the Reynolds number. More details on the computational set-up for the square duct case are provided by McConkey *et al.* (2021). The boundary conditions are periodic at the inlet/outlet, and no-slip walls were applied along the sides of the duct. As discussed in § 1.5.1, the modified TBNN was trained on several values of Re_H, with $Re_H = 2000$ serving as a hold-out test case.

Figure 16 shows the components of the anisotropy tensor a_{ij} predicted by RANS, DNS and the TBNN/KCNN models for the square duct test case. The k-ω SST model is a linear eddy viscosity model, and therefore predicts zero a_{ij} where S_{ij} is zero. Figure 16 shows that a_{11}, a_{22}, a_{23} and a_{33} are all non-zero in the duct, and that the k-ω SST model is unable to capture this behaviour. The TBNN/KCNN models predict an accurate evolution of almost all anisotropy tensor components across the duct cross-section (*viz.* a_{11}, a_{12}, a_{13}, a_{22} and a_{33} are all predicted well on this test case). The anisotropy tensor component a_{23} is not predicted well, likely because it is at least an order of magnitude smaller than the other components, and therefore errors in a_{23} are not penalised as heavily in the loss function.

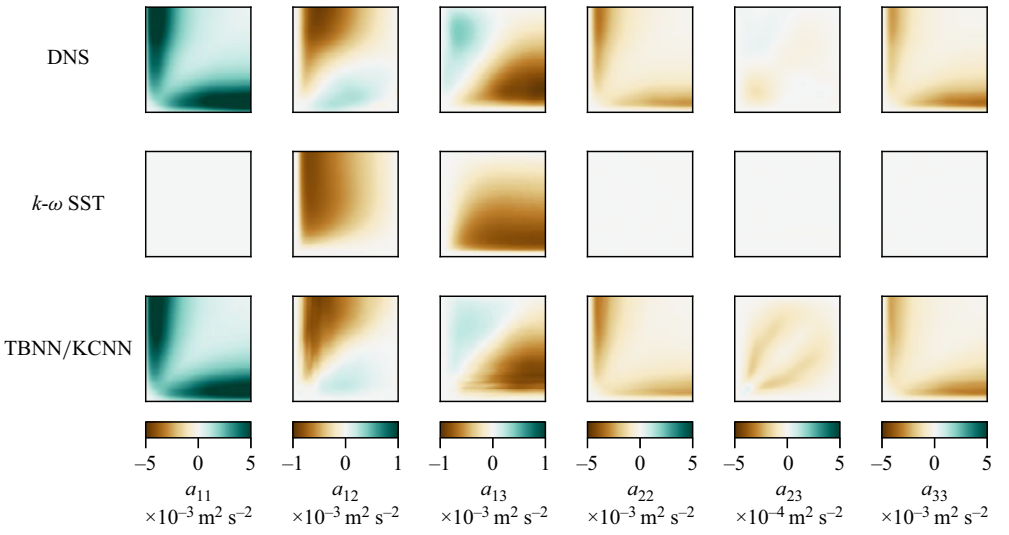


Figure 16. Components of a_{ij} predicted by the DNS data from Pinelli *et al.* (2010), the k-ω SST model, and the TBNN/KCNN model from the present investigation. Shown here are contours of a_{ij} components in the lower left $x_2 \leqslant 0,\ x_3 \leqslant 0$ quadrant.

Figure 17 shows the turbulent kinetic energy k after being corrected by the KCNN model for the square duct test case. Accurate prediction of k is critical to an accurate estimate of a_{ij}, since $a_{ij} = 2kb_{ij}$. The k-ω SST model generally under-predicts k. After correction via the KCNN, the k field is predicted well compared with the DNS data. The primary feature in the k field that is absent from the k-ω SST prediction is the high-k region along the sidewalls of the duct. The KCNN introduces a correction to the baseline RANS field, and is able to predict this high-k region.

Ultimately, it is the goal of the proposed framework to improve the estimated mean fields in the RANS simulation. To determine whether the corrected closure term would produce corrected mean velocity fields, the predicted $\tilde{a}_{ij}^{\perp}$ was injected into the RANS momentum equation as shown in (1.15). The momentum and continuity equations converged around the fixed $\tilde{a}_{ij}^{\perp}$ until numerical convergence was achieved. In OpenFOAM v2212, a modified version of the Pressure implicit splitting of operators - semi implicit method for pressure linked equations (PIMPLE) solver was implemented for the purpose of this injection. The PIMPLE solver was used to incorporate an unsteady term into the system of equations during iteration, to promote stability. Although this unsteady term affects the solution during convergence, the simulation ultimately achieved a steady-state condition, thereby reducing this unsteady term to zero.

Figure 18 shows that the TBNN/KCNN model is able to produce secondary flows after injecting $\tilde{a}_{ij}^{\perp}$ into the momentum equation. This *a posteriori* prediction of the mean field is ultimately the main prediction of interest for a ML-augmented RANS closure framework. Whereas the original k-ω SST model does not predict formation of any secondary flows in the duct, figure 18 shows that the ML-augmented k-ω SST model predicts corner vortices. However, the in-plane kinetic energy is generally underpredicted by the ML-augmented RANS model.

To further examine the ability of the ML-augmented k-ω SST model to predict secondary flows in the duct test case, profiles of U_2 and U_3 are plotted in figure 19. While the ML-augmented model is able to produce this nonlinear feature, the corner

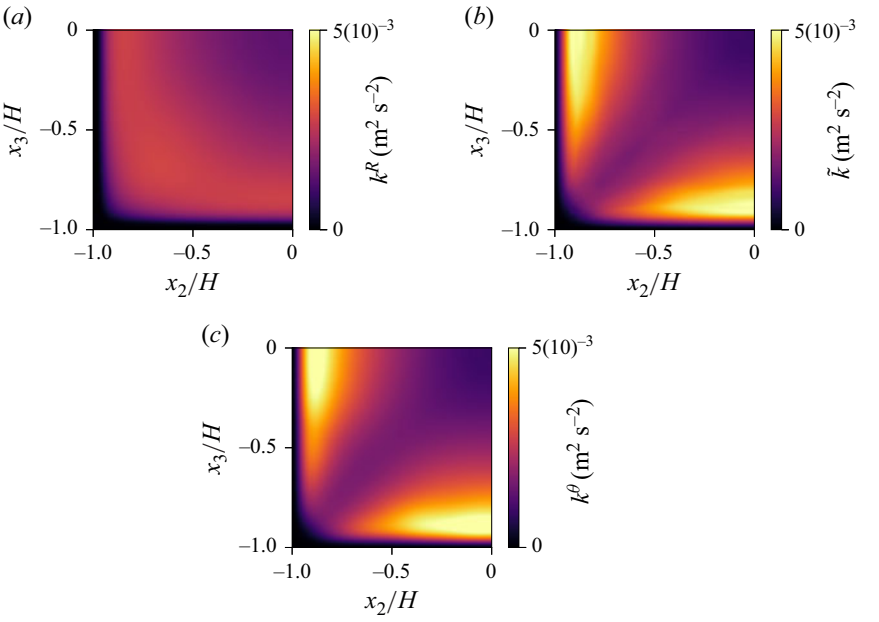


Figure 17. Contours of turbulent kinetic energy k predicted by (a) the k-ω SST model, (b) the KCNN in the present investigation and (c) the DNS data from Pinelli *et al.* (2010). Shown here is the lower left $x_2 \leqslant 0,\ x_3 \leqslant 0$ quadrant.

vortex strengths are reduced compared with the reference DNS data. Both the U_2 and U_3 components are under-predicted. Nevertheless, the k-ω SST model (which predicts $U_2 = U_3 = 0$) has clearly been improved via a ML-augmented correction to the closure term in the momentum equation. From figure 16, it would appear that the good prediction of the normal stress anisotropy (the primary mechanism responsible for the streamwise vorticity determined by U_2 and U_3) should provide good predictions of the streamwise vorticity. However, once the secondary flow is set in motion by this normal stress anisotropy, it is the secondary (rather than primary) shear stress component a_{23} (generated by the presence of the secondary flow itself) that is required to maintain this flow and from figure 16, this secondary component of the shear stress is not well predicted. Therefore, the underprediction of U_2 and U_3 is likely due to inaccurate prediction of a_{23}.

2.1.4. *Rectangular duct test case*

Duct aspect ratio has an influence on the in-plane kinetic energy $1/2(U_2^2 + U_3^2)$ and behaviour of the corner vortices (Vinuesa *et al.* 2014, 2015, 2016). It was of interest to examine the ability of a model trained on square ducts to generalise to a rectangular duct at similar Reynolds number. The TBNN/KCNN models were applied to a duct with aspect ratio 3 and $Re_H \approx 2600$, matching the DNS simulation by Vinuesa, Schlatter & Nagib (2018). This generalisation test was carried out in the same manner as the $Re_H = 2000$ hold-out test case (i.e. by making a predictive correction to the closure terms in the k-ω SST turbulence model, and injecting them back into the momentum equation).

Figure 20 compares the in-plane kinetic energy and velocity vector field predicted by the baseline k-ω SST model, and the TBNN/KCNN augmented k-ω SST model. Examining figure 20 shows that ML-based correction enables the augmented SST model to predict secondary flows. However, in the same manner as the square duct test case, the in-plane kinetic energy is underpredicted. Additionally, the strength of the dominant

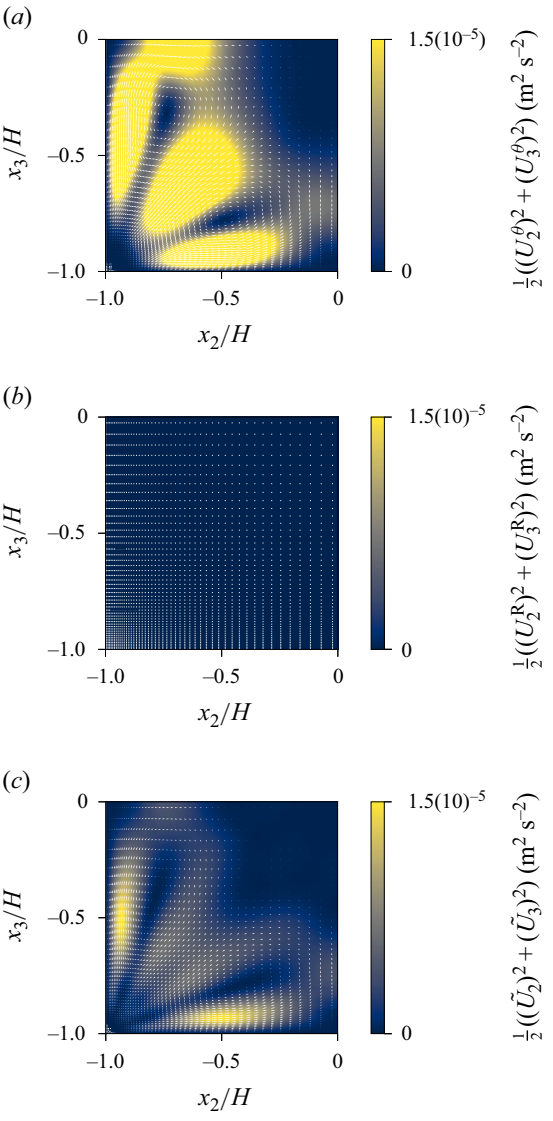


Figure 18. Velocity vectors and in-plane kinetic energy predicted by (*a*) the DNS data from Pinelli *et al.* (2010), (*b*) the k-ω SST model and (*c*) injecting the TBNN/KCNN predictions into the RANS momentum equations. Shown here is the lower left $x_2 \leqslant 0,\ x_3 \leqslant 0$ quadrant.

corner vortex is over-predicted by the ML-augmented model, while the strength of the smaller corner vortex is significantly underpredicted. The magnitude of the in-plane kinetic energy is similar, however, the peak locations of this field are also shifted due to a mismatch in predicted corner vortex shape. Nevertheless, the ML-augmented model shows improvement compared with the k-ω SST model, which fails to predict any corner vortices. In order to promote better generalisation to rectangular geometries, a more extensive training dataset consisting of rectangular ducts would need to be used to train the models.

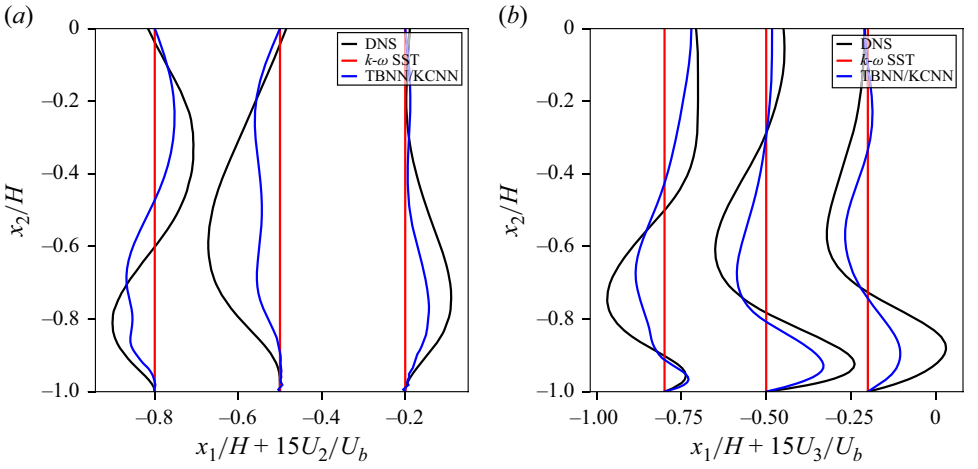


Figure 19. Profiles of the in-plane velocity components (*a*) U_2 and (*b*) U_3, as predicted by the k-ω SST model, the injected TBNN/KCNN predictions and the DNS data from Pinelli *et al.* (2010). Shown here is the lower left $x_2 \leqslant 0,\ x_3 \leqslant 0$ quadrant.

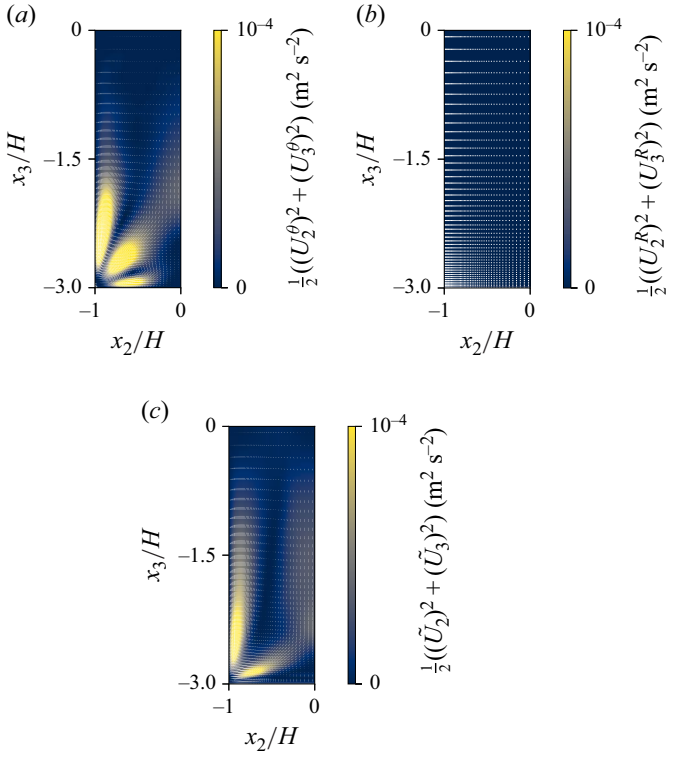


Figure 20. Velocity vectors and in-plane kinetic energy predicted by (*a*) the DNS data from Pinelli *et al.* (2010), (*b*) the k-ω SST model and (*c*) injecting the TBNN/KCNN predictions into the RANS momentum equations. Shown here is the lower left $x_2 \leqslant 0,\ x_3 \leqslant 0$ quadrant.

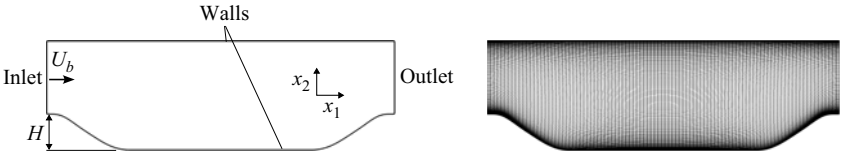


Figure 21. Computational domain and mesh for the periodic hill case, with $\alpha = 1.2$.

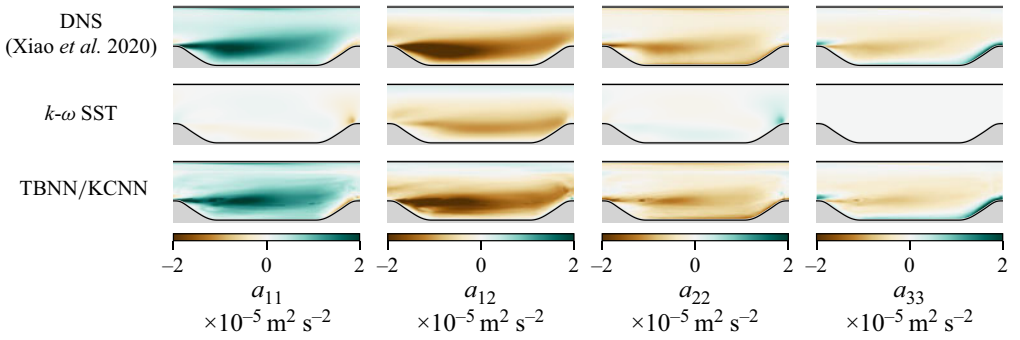


Figure 22. Contours of non-zero a_{ij} components predicted by the DNS data from Xiao *et al.* (2020), the k-ω SST model and the TBNN/KCNN model predictions from the present investigation.

2.1.5. *Periodic hills*

Flow over periodic hills is used as a popular benchmark case for turbulence modelling given the challenging physics of boundary layer separation in an adverse pressure gradient, reattachment along the bottom wall, and acceleration of the flow before re-entering the domain. For the purpose of data-driven turbulence modelling, a variety of periodic hill data have been made available. In this study, we use the $Re_H = 5600$ configuration, simulated using DNS by Xiao *et al.* (2020). Xiao *et al.*'s data were included in McConkey *et al.* (2021), which is the primary data source for this study.

The geometry and mesh for the periodic hill case are shown in figure 21. For all periodic hill cases, the hill height-based Reynolds number is 5600, calculated by

$$Re_H = \frac{U_b H}{\nu}, \tag{2.4}$$

where U_b is the bulk (mean) velocity at the domain inlet. The hill geometry is varied between cases, based on the hill steepness α. Further details on the computational set-up for the baseline RANS periodic hill simulations are provided in McConkey *et al.* (2021). The boundary conditions are periodic at the inlet/outlet, and no-slip walls at the top and bottom of the domain were imposed. As discussed in § 1.5.1, the TBNN and KCNN models were trained on several hill steepness values, with $\alpha = 1.2$ being used as a hold-out test set.

Figure 22 shows the components of the anisotropy tensor a_{ij} predicted by RANS (k-ω SST), DNS and the ML-augmented RANS simulation. The improvement in all components of a_{ij} is clear. The baseline k-ω SST model under-predicts all components, with severe under-prediction of a_{11}, a_{22} and a_{33}. The a_{12} prediction by the k-ω SST model showcases similar trends to the DNS data, but the overall magnitude is lower. However, after correction, key features of all a_{ij} fields are captured when the TBNN/KCNN augment the k-ω SST model. In particular, the higher magnitudes of the diagonal a_{ij} (normal stress) components are captured by the augmented model.

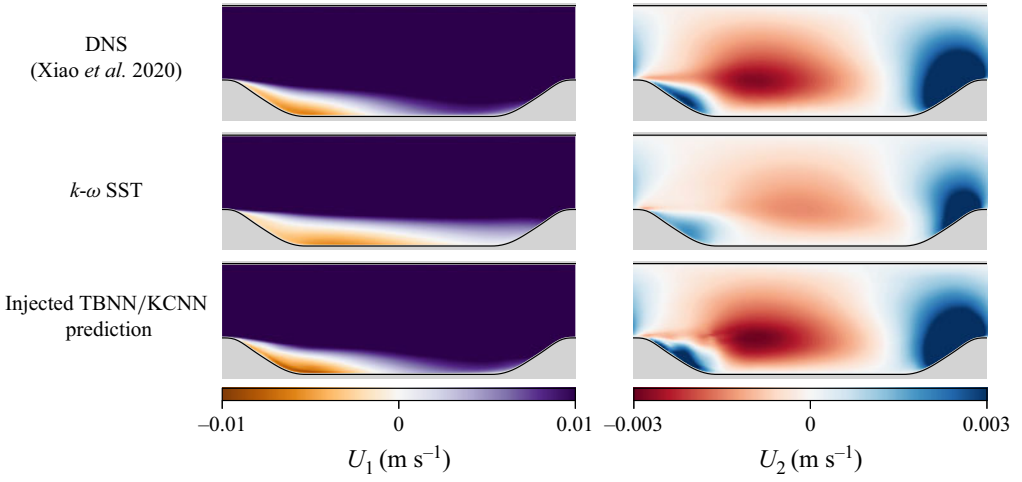


Figure 23. Contours of U_1 and U_1 predicted by the DNS data from Xiao *et al.* (2020), the k-ω SST model and the injected TBNN/KCNN model predictions from the present investigation.

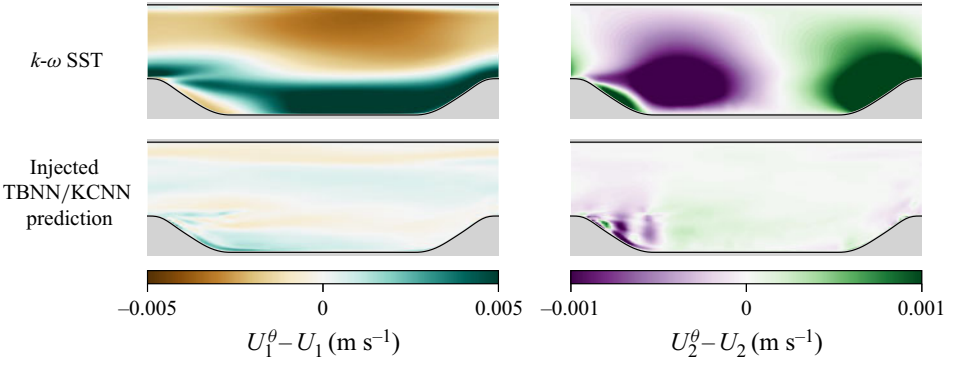


Figure 24. Contours of error in U_1 and U_2 predicted by the k-ω SST model and the injected TBNN/KCNN model predictions, as compared with the DNS data from Xiao *et al.* (2020).

As was done for the square duct test case (§ 2.1.3), a modified PIMPLE solver was used to inject the predicted $\tilde{a}_{ij}$ into the RANS momentum equation for the periodic hill test case. The numerical set-up for the periodic hill injection was identical to the square duct case. Figure 23 compares the mean velocity fields before and after the corrected closure term is used within the RANS simulation. Figure 24 compares the errors in the velocity components U_1 and U_2 estimated by the k-ω SST model, and the *a posteriori* (post-injection) TBNN/KCNN-augmented SST model.

As seen in figure 23, the primary feature of this flow is a recirculation zone which appears immediately after the left hill. The recirculation zone is most clearly visualised by examining the U_1 fields. The k-ω SST model over-predicts the size of this recirculation zone. After correction via injecting the TBNN/KCNN predictions, the recirculation zone size closely matches the DNS data. In the U_2 field, a region with $U_2 < 0$ is seen immediately above this recirculation region. The baseline k-ω SST model under-predicts the downward velocity here, leading to delayed reattachment, and a longer recirculation

Dataset	α	MSE (a_{ij})	% non-realisable
Flat plate	0	$3.5(10)^{-3}$	0.6 %
	10^2	$7.4(10)^{-3}$	0 %
Square duct	0	$2.2(10)^{-9}$	9.2 %
	10^2	$2.6(10)^{-9}$	0.4 %
Periodic hills	0	$3.9(10)^{-13}$	1.7 %
	10^2	$6.3(10)^{-13}$	0.3 %

Table 3. Comparison of mean squared error and number of non-realisable predictions when training with and without a realisability-informed loss function.

zone. After correction, the magnitude of U_2 in this shear layer more closely matches the DNS data. On the right hill, the upward acceleration of the flow under the favourable pressure gradient is also under-predicted by the k-ω SST. Here, the injected $\tilde{a}_{ij}$ is able to better capture the strength of this upward acceleration.

Figure 24 more closely examines the improvements offered by augmenting the k-ω SST model via the TBNN/KCNN. It can be seen that the overall magnitudes of the errors in U_1 and U_2 are significantly reduced after injecting the TBNN/KCNN model predictions. In particular, error in U_1 is reduced in the reattachment region along the bottom wall, and the bulk flow above this region. Error in U_2 is reduced in the previously identified shear layer above the recirculation region, and the accelerating region before the outlet.

2.2. *Impact of realisability-informed training*

To determine the impact that including a realisability-informed penalty has on the training process, the closure term predictions for the three test cases were examined in greater detail. Two loss functions were used: one with $\alpha = 0$ (representing no realisability penalty), and one with $\alpha = 10^2$ (representing an exaggerated realisability penalty term in the loss function). The objective of this test was to determine whether including the realisability penalty during training promotes better generalisation of the closure mapping to unseen flow variations.

All TBNN hyperparameters were fixed to those given in table 2. Since the realisability-informed training procedure only applies to the TBNN, a perfect prediction of k via the KCNN was assumed for calculating error in $\tilde{a}_{ij}$.

Table 3 compares the MSE in $\tilde{a}_{ij}$ on the hold-out test set, with and without realisability penalties being used in training. It should be noted that similar to the *a priori* tests in § 2, the linear component used when visualising b_{ij} comes from S^{θ}_{ij}, as is the configuration when training the TBNN. This linear component is the one used when training the TBNN, and therefore its use here provides the most fair assessment of how the proposed loss function promotes more physically realisable results.

As seen in table 3, the realisability-informed loss function significantly reduces realisability violations on unseen flow variations. Even on hold-out test cases, the model is able to predict b_{ij} without any realisability violations. In some cases, a small tradeoff in a_{ij} occurs – this tradeoff is expected, as for some difficult points the realisability-informed loss function involves a tradeoff between error and realisability. However, in all cases, realisability-informed training also reduces the error in b_{ij}. This error reduction in b_{ij} is expected, since all b_{ij} reference data are realisable. In the case of predicting b_{ij} accurately, the gradients of the realisability penalty $\mathcal{R}(b_{ij})$ further push the predictions towards an accurate prediction of b_{ij}, compared with a purely MSE gradient.

To further visualise the realisability of the TBNN predictions, the barycentric map of Banerjee *et al.* (2007) was used. In the barycentric map, the eigenvalues of a given b_{ij} are mapped into a triangle, the bounds of which represent the limiting realisable behaviours of turbulent fluctuations. This triangle is useful to spatially visualise realisability violations, and types of turbulent flow physics predicted by the baseline turbulence model, the ML-augmented turbulence model and DNS. Further details on the construction of this mapping are given in Banerjee *et al.* (2007) and Emory & Iaccarino (2014).

Figure 25 compares the realisability of the predictions made by a model trained on only a MSE loss function with a model trained on the realisability-informed loss function for the three flows in the present study. All predictions are for the hold-out test set for each flow, representing a generalisation test. The goal of the realisability-informed loss function is to encode a preference for realisable predictions into the model when generalising outside of the training dataset. Figure 25 shows a clear improvement in the realisability of the TBNN predictions. This visualisation supports the results in table 3 in that a realisability-informed model has significantly lower realisability violations when predicting the anisotropy tensor on a new flow. Nearly all of the predictions from the realisability-informed model fall within the realisable boundaries, while the model trained only on MSE predicts several realisability-violating results when generalising to new cases of complex flows such as the duct and periodic hill cases. Also, the violations of physical realisability for the realisability-informed model (when they do occur) are not as severe (as measured from the magnitude of deviation outside the barycentric map) as those obtained from only a MSE loss function.

Figure 25 also shows that for all flows in the present study, the original k-ω SST model predicts plane-strain turbulence. All flows in the present study have a strain rate tensor which results in at least one zero eigenvalue of b_{ij}, for the linear eddy viscosity approximation ($b_{ij} = -\nu_t / k S_{ij}$), and therefore plane-strain turbulence is predicted for all flows by the k-ω SST model.

As discussed in § 1.3, realisability-informed learning function does not guarantee a fully realisable prediction. Rather, realisability-informed learning encodes a preference for realisable predictions into the model predictions. This avoids the need for *ad hoc* post-processing of the predicted anisotropy tensor, while also encoding a physics-based (learning) bias into the TBNN. Strict realisability can be further enforced by post-processing any non-realisable predictions by the TBNN, as the anisotropy tensor can still be accessed and assessed for realisability in the proposed framework (albeit, with an evolving linear component).

An important distinction between the dimensional and non-dimensional anisotropy tensor can also be drawn from the results in this section. In turbulence modelling, it is often the non-dimensional anisotropy tensor b_{ij} that is thought to be of interest – indeed, it is possible to non-dimensionalise the closure problem (e.g. the generalised eddy viscosity model proposed by Pope (1975)). However, in the present study, we show that even with physically realisable DNS data, dimensionalising the anisotropy tensor via $a_{ij} = 2kb_{ij}$ provides a distinct learning target and loss landscape. The realisability-informed loss function does not, in principle, compete against a simple error-based loss function in terms of predicting the non-dimensional b_{ij} – non-realisable predictions will also have high error. However, setting the dimensional anisotropy tensor as the target creates a tradeoff between these two objectives, as demonstrated by the results in table 3. Since predicting the dimensional anisotropy tensor is favourable for the reasons outlined in § 1.3, further investigation is required to determine the exact source of this tradeoff. This future research area is particularly relevant for learning anisotropy mappings from data, a major area of focus for improving RANS via ML.

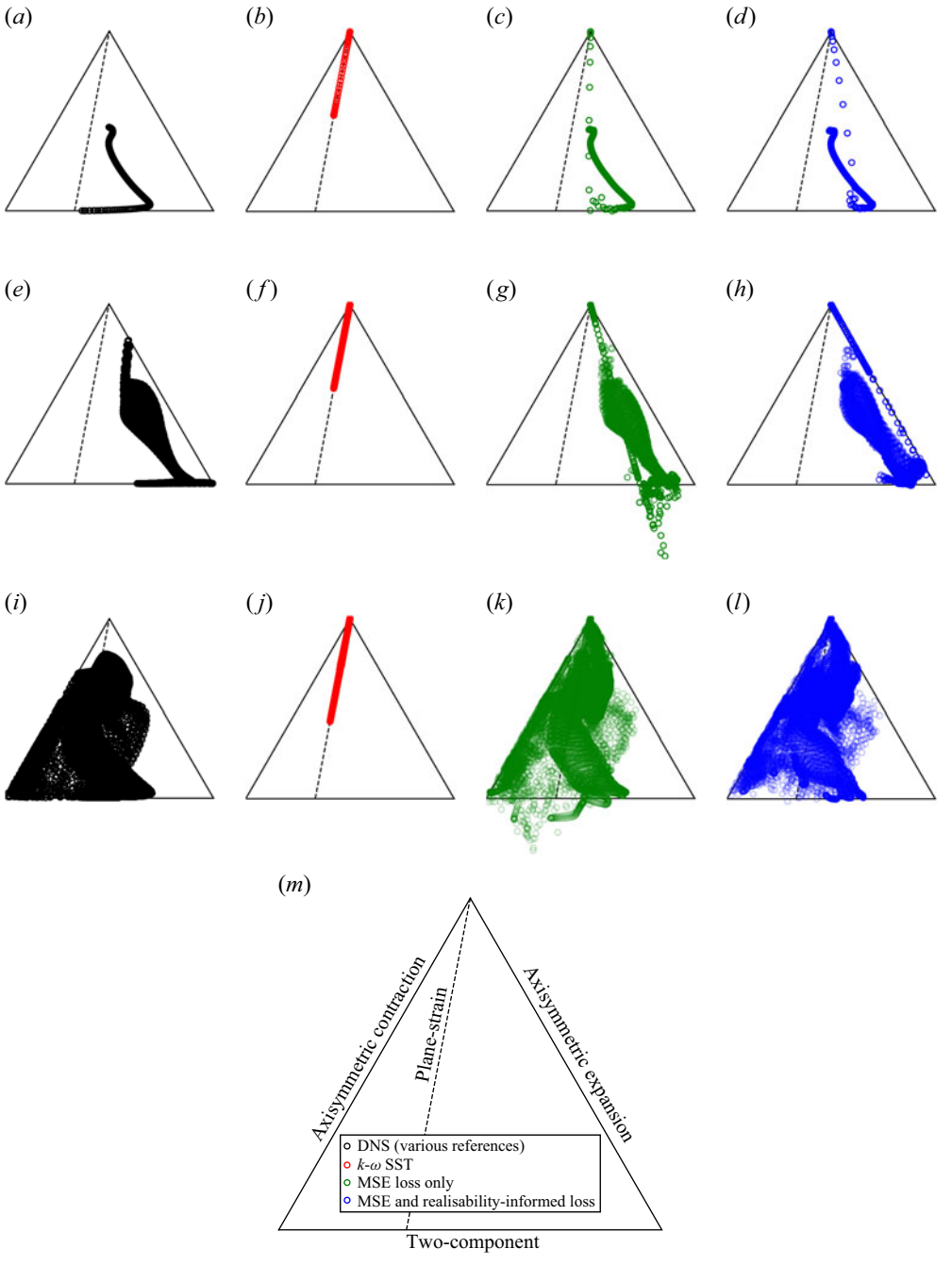


Figure 25. Projection of the predicted b_{ij} from the DNS reference data (*a*,*e*,*i*), the k-ω SST model (*b*, *f*, *j*) and the TBNN/KCNN with (*d*,*h*,*l*) and without (*c*,*g*,*k*) realisability-informed learning onto the barycentric triangle. Panels (*a*)–(*d*) show the flat plate data, (*e*)–(*h*) show the square duct data and (*i*)–(*l*) show the periodic hill data. Points outside the triangle are not realisable.

3. Conclusion

The objectives of this study were to propose a physics-informed loss function for training TBNN anisotropy mappings, a new TBNN architecture and a new injection framework that accommodates implicit treatment of the linear anisotropy component within a TBNN-type architecture. This framework addresses an issue related to the stability of injected TBNN predictions, an issue that has been reported by Kaandorp & Dwight (2020), and consistent with our own experience. This framework also addresses the issue of realisability within the context of predicting the anisotropy tensor from RANS input features.

The results here indicate that the proposed model architecture generalises well to new flow configurations, and the predicted anisotropy tensor can be injected in a highly stable manner. During the injection procedure in the present investigation, the Computational Fluid Dynamics solver remained stable, even when testing erratic model predictions. While this finding corroborates findings from others that frameworks which leverage implicit treatment of the linear anisotropy component via an eddy viscosity are numerically stable (Wu *et al.* 2019*b*; Brener *et al.* 2021; Liu *et al.* 2021), a novelty in the present investigation is the avoidance of using an optimal eddy viscosity. With the proposed decomposition of a_{ij} (§ 1.1), a simpler (and more straightforward) baseline k-ω SST eddy viscosity is used. The core modification that allows a well-conditioned solution in this case is the use of S_{ij}^{θ} within $\hat{T}_{ij}^{(1)}$ in the TBNN training process. As a result, all corrections induced by the TBNN are contained in an explicit term in the momentum equation. Future work includes investigating how new flow generalisation can be improved by a blending factor that multiplies this explicit term, thereby allowing the correction to be turned off. For example, a statistics-based scalar or a separate machine learning model could be used to predict a blending factor. This blending factor could reduce or eliminate corrections when a test data point departs significantly (is out of distribution) from the training dataset, and erroneous predictions are likely.

While the realisability-informed loss function does not strictly guarantee physical realisability of the predictions, the results in § 2.2 indicate that the model retains a realisability bias when generalising. The realisability-informed loss function is not only applicable to the framework and architecture proposed in this study – it could be used anytime an anisotropy mapping is generated via machine learning. The use of the realisability-informed loss function in the present investigation promoted better realisability of predictions by a TBNN, but we fully expect the bias induced by this loss function to also be beneficial for tensor basis random forests (e.g. Kaandorp & Dwight (2020)), or non-tensor basis frameworks (e.g. Wu *et al.* (2018)). Since high-quality DNS anisotropy tensor data are realisable, the realisability penalty can be viewed as an additional boost to the loss function gradient towards the true value, when a non-realisable prediction is made. Future work will investigate how the realisability-informed loss function performs with training data generated by LES, which are not guaranteed to be realisable.

The KCNN used in the proposed framework to correct k could also be replaced by coupling the k equation to the momentum equation, as exists in the original turbulence model, and the TBNN approach originally proposed by Ling *et al.* (2016). At training time, k^{θ} is used to dimensionalise b_{ij} in the present study, since k is corrected via the KCNN. This direct correction produced satisfactory results in the present work, but it is possible that generalisation could be further enhanced by the re-coupling the k equation with an updated closure term. This enhanced generalisation would result from the fact that a physics-based coupled equation system is used to correct k, rather than a simple multiplicative corrector (the KCNN in the present investigation). However, this coupling of an additional partial differential equation introduces the possibility for instability, an issue

which is common in machine learning anisotropy modelling. Future work will investigate the merits of this route.

Ultimately, this investigation demonstrates that with sufficient modifications, TBNN-type anisotropy mappings can be injected in a stable and well-conditioned manner. Further, with appropriate physics-based loss function penalties, the mapping can be sensitised to more physically informative targets than MSE. Moreover, we provided a preliminary investigation of the utility of KANs for turbulence closure modelling. While TBKAN did not outperform TBNN in this context, they nevertheless demonstrated the potential for capturing the complex relationships in the anisotropy tensor (at least in the flat plate case), suggesting that future research work should be conducted on the use of KANs for turbulent closure modelling applications. While industrial use of machine learning-based anisotropy mappings is currently not widespread, the continual development of techniques which increase the practicality of training and injecting model predictions will help accelerate more widespread use. Machine learning-augmented turbulence closure modelling is an aid that the turbulence modelling community can use to help bridge the current computational gap between RANS and widespread use of LES (Witherden & Jameson 2017).

Acknowledgements. We sincerely thank the anonymous reviewers for their careful review and helpful suggestions for the manuscript.

Funding. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarship program. Computational resources for this work were supported by the Tyler Lewis Clean Energy Research Foundation.

Declaration of interests. The authors report no conflict of interest.

Data availability statement. All code and data that support the results of this study are publicly available. The code can be found at https://github.com/rmcconke/tbnn. The dataset used in this work is available at https://doi.org/10.34740/kaggle/dsv/2637500.

Author contributions. R.M.: conceptualisation, methodology, software, validation, investigation, data curation, writing (original draft, review and editing), visualisation. N.K. (KAN related work): conceptualisation, methodology, software, validation, investigation, writing (original draft, review and editing), visualisation. E.Y.: conceptualisation, methodology writing (review and editing), supervision, project administration, funding acquisition. F.S.L.: conceptualisation, methodology, writing (review and editing), supervision, project administration, funding acquisition.

Appendix A. Integrity basis input features

Wu *et al.*'s integrity basis is derived from four gradient tensors: $\hat{S}$, $\hat{R}$, $\hat{A}_p$ and $\hat{A}_k$ (Wu *et al.* 2018). These tensors are calculated as follows:

$$\hat{S}_{ij} = C_S \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right), \tag{A1}$$

$$\hat{R}_{ij} = C_R \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right), \tag{A2}$$

$$\hat{A}_p = C_{A_p} \epsilon_{ijl} \frac{\partial p}{\partial x_l}, \tag{A3}$$

$$\hat{A}_k = C_{A_k} \epsilon_{ijl} \frac{\partial k}{\partial x_l}, \tag{A4}$$

where C_S, C_R, C_{A_p} and C_{A_k} are scalars which non-dimensionalise their corresponding gradient tensors, and ϵ_{ijl} is the Levi-Civita symbol. For example, Ling *et al.* (2016) chose $C_S = C_R = k/\varepsilon$, so that $\hat{S}$ is dimensionless.

Without loss of generality, the scalars p and k in the above can be swapped out for other scalars, such as ε, or ω, thereby replacing $\hat{A}_{p,ij}$ and $\hat{A}_{k,ij}$ with $\hat{C}_{ij}$ and $\hat{D}_{ij}$

$$\hat{C}_{ij} = C_C \epsilon_{ijl} \frac{\partial s_1}{\partial x_l}, \tag{A5}$$

$$\hat{D}_{ij} = C_D \epsilon_{ijl} \frac{\partial s_2}{\partial x_l}, \tag{A6}$$

where s_1 and s_2 are two scalar fields. For example, in the present study, $s_1 = \omega$, and $s_2 = k$. In 3-D Cartesian coordinates, the strain rate, rotation rate and antisymmetric tensors associated with the two scalar gradients are

$$\hat{S}_{ij} = \frac{C_S}{2} \begin{bmatrix} 2\dfrac{\partial U_1}{\partial x_1} & \dfrac{\partial U_1}{\partial x_2} + \dfrac{\partial U_2}{\partial x_1} & \dfrac{\partial U_1}{\partial x_3} + \dfrac{\partial U_3}{\partial x_1} \\ \dfrac{\partial U_2}{\partial x_1} + \dfrac{\partial U_1}{\partial x_2} & 2\dfrac{\partial U_2}{\partial x_2} & \dfrac{\partial U_2}{\partial x_3} + \dfrac{\partial U_3}{\partial x_2} \\ \dfrac{\partial U_3}{\partial x_1} + \dfrac{\partial U_1}{\partial x_3} & \dfrac{\partial U_3}{\partial x_2} + \dfrac{\partial U_2}{\partial x_3} & 2\dfrac{\partial U_3}{\partial x_3} \end{bmatrix}, \tag{A7}$$

$$\hat{R}_{ij} = \frac{C_R}{2} \begin{bmatrix} 0 & \dfrac{\partial U_1}{\partial x_2} - \dfrac{\partial U_2}{\partial x_1} & \dfrac{\partial U_1}{\partial x_3} - \dfrac{\partial U_3}{\partial x_1} \\ \dfrac{\partial U_2}{\partial x_1} - \dfrac{\partial U_1}{\partial x_2} & 0 & \dfrac{\partial U_2}{\partial x_3} - \dfrac{\partial U_3}{\partial x_2} \\ \dfrac{\partial U_3}{\partial x_1} - \dfrac{\partial U_1}{\partial x_3} & \dfrac{\partial U_3}{\partial x_2} - \dfrac{\partial U_2}{\partial x_3} & 0 \end{bmatrix}, \tag{A8}$$

$$\hat{C}_{ij} = C_C \begin{bmatrix} 0 & -\dfrac{\partial s_1}{\partial x_3} & \dfrac{\partial s_1}{\partial x_2} \\ \dfrac{\partial s_1}{\partial x_3} & 0 & -\dfrac{\partial s_1}{\partial x_1} \\ -\dfrac{\partial s_1}{\partial x_2} & \dfrac{\partial s_1}{\partial x_1} & 0 \end{bmatrix}, \tag{A9}$$

$$\hat{D}_{ij} = C_D \begin{bmatrix} 0 & -\dfrac{\partial s_2}{\partial x_3} & \dfrac{\partial s_2}{\partial x_2} \\ \dfrac{\partial s_2}{\partial x_3} & 0 & -\dfrac{\partial s_2}{\partial x_1} \\ -\dfrac{\partial s_2}{\partial x_2} & \dfrac{\partial s_2}{\partial x_1} & 0 \end{bmatrix}. \tag{A10}$$

Under some conditions, input features derived from invariants of the minimal integrity basis derived by Wu *et al.* (2018) can vanish. These conditions occur when there are zero gradients in the flow, as all tensors in Wu's *et al.*'s integrity basis are derived from gradient-based tensors.

Here, we consider two cases.

(i) Two-dimensional flow.
The zero pressure gradient boundary layer and periodic hill cases in the present study fall into this category. With the coordinate system defined as it was in § 2.1.5, the

velocity vector is $U_j = (U_1, U_2, 0)$, and the gradient tensors take the following form in 3-D space:

$$\hat{S}_{ij} = \frac{C_S}{2}\begin{bmatrix} 2\dfrac{\partial U_1}{\partial x_1} & \dfrac{\partial U_1}{\partial x_2} + \dfrac{\partial U_2}{\partial x_1} & 0 \\ \dfrac{\partial U_2}{\partial x_1} + \dfrac{\partial U_1}{\partial x_2} & 2\dfrac{\partial U_2}{\partial x_2} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{A11}$$

$$\hat{R}_{ij} = \frac{C_R}{2}\begin{bmatrix} 0 & \dfrac{\partial U_1}{\partial x_2} - \dfrac{\partial U_2}{\partial x_1} & 0 \\ \dfrac{\partial U_2}{\partial x_1} - \dfrac{\partial U_1}{\partial x_2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{A12}$$

$$\hat{C}_{ij} = C_C\begin{bmatrix} 0 & 0 & \dfrac{\partial s_1}{\partial x_2} \\ 0 & 0 & -\dfrac{\partial s_1}{\partial x_1} \\ -\dfrac{\partial s_1}{\partial x_2} & \dfrac{\partial s_1}{\partial x_1} & 0 \end{bmatrix}, \tag{A13}$$

$$\hat{D}_{ij} = C_D\begin{bmatrix} 0 & 0 & \dfrac{\partial s_2}{\partial x_2} \\ 0 & 0 & -\dfrac{\partial s_2}{\partial x_1} \\ -\dfrac{\partial s_2}{\partial x_2} & \dfrac{\partial s_2}{\partial x_1} & 0 \end{bmatrix}. \tag{A14}$$

(ii) Three-dimensional flow with zero gradients in one direction.
The square duct case in the present study falls into this category. With the coordinate system defined as it was in § 2.1.3, the velocity vector is $U_j = (U_1, U_2, U_3)$. All gradients in the x_1 direction are zero: $\partial()/\partial x_1 = 0$. In this case, the gradient tensors take the following form in 3-D space:

$$\hat{S}_{ij} = \frac{C_S}{2}\begin{bmatrix} 0 & \dfrac{\partial U_1}{\partial x_2} & \dfrac{\partial U_1}{\partial x_3} \\ \dfrac{\partial U_1}{\partial x_2} & 2\dfrac{\partial U_2}{\partial x_2} & \dfrac{\partial U_2}{\partial x_3} + \dfrac{\partial U_3}{\partial x_2} \\ \dfrac{\partial U_1}{\partial x_3} & \dfrac{\partial U_3}{\partial x_2} + \dfrac{\partial U_2}{\partial x_3} & 2\dfrac{\partial U_3}{\partial x_3} \end{bmatrix}, \tag{A15}$$

$$\hat{R}_{ij} = \frac{C_R}{2}\begin{bmatrix} 0 & \dfrac{\partial U_1}{\partial x_2} & \dfrac{\partial U_1}{\partial x_3} \\ \dfrac{\partial U_1}{\partial x_2} & 0 & \dfrac{\partial U_2}{\partial x_3} - \dfrac{\partial U_3}{\partial x_2} \\ \dfrac{\partial U_1}{\partial x_3} & \dfrac{\partial U_3}{\partial x_2} - \dfrac{\partial U_2}{\partial x_3} & 0 \end{bmatrix}, \tag{A16}$$

$$\hat{C}_{ij} = C_C \begin{bmatrix} 0 & -\dfrac{\partial s_1}{\partial x_3} & \dfrac{\partial s_1}{\partial x_2} \\ \dfrac{\partial s_1}{\partial x_3} & 0 & 0 \\ -\dfrac{\partial s_1}{\partial x_2} & 0 & 0 \end{bmatrix}, \tag{A17}$$

$$\hat{D}_{ij} = C_D \begin{bmatrix} 0 & -\dfrac{\partial s_2}{\partial x_3} & \dfrac{\partial s_2}{\partial x_2} \\ \dfrac{\partial s_2}{\partial x_3} & 0 & 0 \\ -\dfrac{\partial s_2}{\partial x_2} & 0 & 0 \end{bmatrix}. \tag{A18}$$

Cases **(i)** and **(ii)** were analysed using sympy (Meurer *et al.* 2017) to determine which integrity basis tensor invariants are non-zero, and therefore suitable as potential input features. The first and second invariants of a rank two tensor are scalar functions, defined by

$$I_1(A_{ij}) = A_{ii}, \tag{A19}$$

$$I_2(A_{ij}) = \frac{1}{2}\left((A_{ii})^2 - A_{ij}A_{ji}\right). \tag{A20}$$

The third invariant, $I_3 = \mathbf{det}(A_{ij})$ is zero for all of the integrity basis tensors, since they are either antisymmetric, or symmetric and zero trace. Table 4 shows the results of this analysis.

Source code which supports the analysis in this appendix and enables further investigation is available on Github (McConkey 2023).

Tensor	Expression	(I)$I_1 \neq 0$	(I)$I_2 \neq 0$	(II)$I_1 \neq 0$	(II)$I_2 \neq 0$
$B_{ij}^{(1)}$	$\hat{S}_{ik}\hat{S}_{kj}$	✓	✓	✓	✓
$B_{ij}^{(2)}$	$\hat{S}_{ik}\hat{S}_{kl}\hat{S}_{lj}$	—	✓	—	✓
$B_{ij}^{(3)}$	$\hat{R}_{ik}\hat{R}_{kj}$	✓	✓	✓	✓
$B_{ij}^{(4)}$	$\hat{C}_{ik}\hat{C}_{kj}$	✓	✓	✓	✓
$B_{ij}^{(5)}$	$\hat{D}_{ik}\hat{D}_{kj}$	✓	✓	✓	✓
$B_{ij}^{(6)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lj}$	—	✓	—	✓
$B_{ij}^{(7)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	✓	✓	✓	✓
$B_{ij}^{(8)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{R}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	✓	—	✓
$B_{ij}^{(9)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{S}_{lj}$	✓	—	—	✓
$B_{ij}^{(10)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(11)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{C}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	✓	✓
$B_{ij}^{(12)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{S}_{lj}$	✓	—	—	✓
$B_{ij}^{(13)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	✓	—	✓	✓

Table 4. For caption see next page.

Tensor	Expression	(I)$I_1 \neq 0$	(I)$I_2 \neq 0$	(II)$I_1 \neq 0$	(II)$I_2 \neq 0$
$B_{ij}^{(14)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{D}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	✓	✓
$B_{ij}^{(15)}$	$\hat{R}_{ik}\hat{C}_{kj}$	—	—	✓	✓
$B_{ij}^{(16)}$	$\hat{C}_{ik}\hat{D}_{kj}$	✓	✓	✓	✓
$B_{ij}^{(17)}$	$\hat{R}_{ik}\hat{D}_{kj}$	—	—	✓	✓
$B_{ij}^{(18)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{S}_{lj}$	—	—	—	✓
$B_{ij}^{(19)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	—	—	✓	✓
$B_{ij}^{(20)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{C}_{lm}\hat{S}_{mj}$	—	—	—	✓
$B_{ij}^{(21)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{R}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(22)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{C}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(23)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{R}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(24)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{C}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	—	✓
$B_{ij}^{(25)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{R}_{mn}\hat{S}_{no}\hat{S}_{oj}$	✓	—	✓	✓
$B_{ij}^{(26)}$	$\hat{R}_{ik}\hat{D}_{kl}\hat{S}_{lj}$	—	—	—	✓
$B_{ij}^{(27)}$	$\hat{R}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	—	—	✓	✓
$B_{ij}^{(28)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{D}_{lm}\hat{S}_{mj}$	—	—	—	✓
$B_{ij}^{(29)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{R}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(30)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{D}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(31)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{R}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(32)}$	$\hat{R}_{ik}\hat{R}_{kl}\hat{S}_{lm}\hat{D}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	—	✓
$B_{ij}^{(33)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{R}_{mn}\hat{S}_{no}\hat{S}_{oj}$	✓	—	✓	✓
$B_{ij}^{(34)}$	$\hat{C}_{ik}\hat{D}_{kl}\hat{S}_{lj}$	✓	—	—	✓
$B_{ij}^{(35)}$	$\hat{C}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(36)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{D}_{lm}\hat{S}_{mj}$	—	—	✓	✓
$B_{ij}^{(37)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{C}_{lm}\hat{S}_{mj}$	—	—	✓	✓
$B_{ij}^{(38)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{D}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(39)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{C}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	—	—	—	✓
$B_{ij}^{(40)}$	$\hat{C}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{D}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	✓	✓
$B_{ij}^{(41)}$	$\hat{D}_{ik}\hat{D}_{kl}\hat{S}_{lm}\hat{C}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	✓	✓
$B_{ij}^{(42)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{D}_{lj}$	✓	—	—	✓
$B_{ij}^{(43)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{D}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(44)}$	$\hat{R}_{ik}\hat{D}_{kl}\hat{C}_{lm}\hat{S}_{mj}$	✓	—	✓	✓
$B_{ij}^{(45)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{D}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	✓	—	—	✓
$B_{ij}^{(46)}$	$\hat{R}_{ik}\hat{D}_{kl}\hat{C}_{lm}\hat{S}_{mn}\hat{S}_{nj}$	✓	—	—	✓
$B_{ij}^{(47)}$	$\hat{R}_{ik}\hat{C}_{kl}\hat{S}_{lm}\hat{D}_{mn}\hat{S}_{no}\hat{S}_{oj}$	—	—	—	✓

Table 4 (cntd). Non-zero invariants of the minimal integrity basis tensor formed by $\hat{S}_{ij}$, $\hat{R}_{ij}$, $\hat{C}_{ij}$ and $\hat{D}_{ij}$.

References

Amarloo, A., Forooghi, P. & Abkar, M. 2022 Frozen propagation of the Reynolds force vector from high-fidelity data into the Reynolds-averaged simulations of secondary flows, 1–16.

Banerjee, S., Krahl, R., Durst, F. & Zenger, C. 2007 Presentation of anisotropy properties of turbulence, invariants versus eigenvalue approaches. *J. Turbul.* **8**, 1–27.

Bobke, A., Vinuesa, R., Örlü, R. & Schlatter, P. 2017 History effects and near equilibrium in adverse-pressure-gradient turbulent boundary layers. *J. Fluid Mech.* **820**, 667–692.

BRENER, B.P., CRUZ, M.A., MACEDO, M.S.S. & THOMPSON, R.L. 2022 An invariant and highly–accurate strategy for data-driven turbulence modelling. *SSRN Electron. J.* 1–43. https://doi.org/10.2139/ssrn.4073177

BRENER, B.P., CRUZ, M.A., THOMPSON, R.L. & ANJOS, R.P. 2021 Conditioning and accurate solutions of Reynolds average Navier–Stokes equations with data-driven turbulence closures. *J. Fluid Mech.* **915**, 1–27.

BRUNTON, S.L., NOACK, B.R. & KOUMOUTSAKOS, P. 2020 Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52** (1), 1–32.

CAI, J., ANGELI, P.-E., MARTINEZ, J.-M., DAMBLIN, G. & LUCOR, D. 2022 Reynolds stress anisotropy tensor predictions for turbulent channel flow using neural networks. https://doi.org/10.48550/arXiv.2208.14301

CAI, J., ANGELI, P.-E., MARTINEZ, J.-M., DAMBLIN, G. & LUCOR, D. 2024 Revisiting tensor basis neural networks for Reynolds stress modeling: application to plane channel and square duct flows. *Comput. Fluids* **275**, 106246.

CAYLEY, A. 1858 II. A memoir on the theory of matrices. *Phil. Trans. R. Soc. Lond.* **148**, 17–37.

CRUZ, M.A., THOMPSON, R.L., SAMPAIO, L.E.B. & BACCHI, R.D.A. 2019 The use of the Reynolds force vector in a physics informed machine learning approach for predictive turbulence modeling. *Comput. Fluids* **192**, 104258.

DURAISAMY K. 2021 Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. *Phys. Rev. Fluids* **6**, 050504.

DURAISAMY, K., IACCARINO, G. & XIAO, H. 2019 Turbulence modeling in the age of data. *Annu. Rev. Fluid Mech.* **51** (1), 357–377.

EMORY, M. & IACCARINO, D.G. 2014 *Visualizing Turbulence Anisotropy in the Spatial Domain with Componentality Contours*. Center for Turbulence Research Annual Research Briefs.

HAMILTON, W.R. 1853 *Lectures On Quaternions*. Hodges and Smith.

JIANG, C., VINUESA, R., CHEN, R., MI, J., LAIMA, S. & LI, H. 2021 An interpretable framework of data-driven turbulence modeling using deep neural networks. *Phys. Fluids* **33** (5), 055133.

KAANDORP, M. 2018 *Machine Learning for Data-Driven RANS Turbulence Modelling*. Delft University of Technology.

KAANDORP, M.L.A. & DWIGHT, R.P. 2020 Data-driven modelling of the Reynolds stress tensor using random forests with invariance. *Comput. Fluids* **202**, 104497.

KINGMA, D.P. & BA, J. 2015 Adam: a method for stochastic optimization. In *ICLR 2015*. https://doi.org/10.48550/arXiv.1412.6980

KOCHKOV, D., SMITH, J.A., ALIEVA, A., WANG, Q., BRENNER, M.P. & HOYER, S. 2021 Machine learning–accelerated computational fluid dynamics. *Proc. Natl Acad. Sci. USA* **118** (21), e2101784118.

LAUNDER, B.E. & SPALDING, D.B. 1974 The numerical computation of turbulent flows. *Comput. Meth. Appl. Mech. Engng* **3** (2), 269–289.

LING, J., KURZAWSKI, A. & TEMPLETON, J. 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **807**, 155–166.

LIST, B., CHEN, L.-W. & THUEREY, N. 2022 Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *J. Fluid Mech.* **949**, A25.

LIU, W., FANG, J., ROLFO, S., MOULINEC, C. & EMERSON, D.R. 2021 An iterative machine-learning framework for RANS turbulence modeling. *Intl J. Heat Fluid Flow* **90**, 108822.

LIU, Z., WANG, Y., VAIDYA, S., RUEHLE, F., HALVERSON, J., SOLJAČIĆ, M., HOU, T.Y. 2024 Kan: Kolmogorov–Arnold networks, arXiv preprint arXiv:2404.19756.

MAN, A., JADIDI, M., KESHMIRI, A., YIN, H. & MAHMOUDI, Y. 2023 A divide-and-conquer machine learning approach for modeling turbulent flows. *Phys. Fluids* **35** (5), 055110.

MANDLER, H. & WEIGAND, B. 2023 Feature importance in neural networks as a means of interpretation for data-driven turbulence models. *Comput. Fluids* **265**, 105993.

MATAI, R. & DURBIN, P.A. 2019 Zonal Eddy viscosity models based on machine learning. *Flow Turbul. Combust.* **103** (1), 93–109.

MCCONKEY, R. 2023 Integrity-basis-input-features. Available at: https://github.com/rmcconke/integrity-basis-input-features.

MCCONKEY, R. & KALIA, N. 2024 TBNN. Available at: https://github.com/rmcconke/tbnn.

MCCONKEY, R., YEE, E. & LIEN, F.-S. 2021 A curated dataset for data-driven turbulence modelling. *Sci. Data* **8** (1), 1–14.

MCCONKEY, R., YEE, E. & LIEN, F.-S. 2022*a* Deep structured neural networks for turbulence closure modeling. *Phys. Fluids* **34** (3), 035110.

MCCONKEY, R., YEE, E. & LIEN, F.-S. 2022*b* On the Generalizability of machine-learning-assisted anisotropy mappings for predictive turbulence modelling. *Intl J. Comput. Fluid Dyn.* **36** (7), 555–577.

MENTER, F.R. 1994 Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J.* **32** (8), 1598–1605.
MENTER, F.R., KUNTZ, M. & LANGTRY, R. 2003 Ten years of industrial experience with the SST turbulence model. *Turbul. Heat Mass Transfer* **4**, 625–632.
MEURER, A. *et al.* 2017 SymPy: symbolic computing in Python. *PeerJ Comput. Sci.* **3**, e103.
NIKITIN, N.V., POPELENSKAYA, N.V. & STROH, A. 2021 Prandtl's secondary flows of the second kind. Problems of description, prediction, and simulation. *Fluid Dyn.* **56** (4), 513–538.
PINELLI, A., UHLMANN, M., SEKIMOTO, A. & KAWAHARA, G. 2010 Reynolds number dependence of mean flow structure in square duct turbulence. *J. Fluid Mech.* **644**, 107–122.
POPE, S.B. 1975 A more general effective-viscosity hypothesis. *J. Fluid Mech.* **72** (02), 331.
POPE, S.B. 2000 *Turbulent Flows*. Cornell University.
RAISSI, M., PERDIKARIS, P. & KARNIADAKIS, G.E. 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707.
RAMACHANDRAN, P., ZOPH, B. & LE, Q.V. 2017 Searching for activation functions. arXiv.1710.05941.
REDDI, S.J., KALE, S. & KUMAR, S. 2018 On the convergence of Adam and beyond. arXiv:1904.09237.
REYNOLDS, O. 1895 IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Phil. Trans. R. Soc. Lond. A* **186**, 123–164.
RICCIUS, L., AGRAWAL, A. & KOUTSOURELAKIS, P.-S. 2023 Physics-informed tensor basis neural network for turbulence closure modeling, arXiv:2311.14576.
RUMSEY, C. 2021 NASA langley research center turbulence modeling resource. Available at: https://turbmodels.larc.nasa.gov/.
SCHLATTER, P. & ÖRLÜ, R. 2010 Assessment of direct numerical simulation data of turbulent boundary layers. *J. Fluid Mech.* **659**, 116–126.
SINGH, A.P., DURAISAMY, K. & ZHANG, Z.J. 2017 Augmentation of Turbulence Models Using Field Inversion and Machine Learning. In 55th AIAA Aerospace Sciences Meeting. American Institute of Aeronautics and Astronautics.
SIRIGNANO, J. & MACART, J.F. 2023 Deep learning closure models for large-eddy simulation of flows around bluff bodies. *J. Fluid Mech.* **966**, A26.
SLOTNICK, J., KHODADOUST, A., ALONSO, J., DARMOFAL, D., GROPP, W., LURIE, E. & MARVRIPLIS, D. 2014 CFD Vision 2030 Study: a path to revolutionary computational aerosciences. *Tech. Rep.* NASA/CR-2014-218178. Available at: https://ntrs.nasa.gov/citations/20140003093.
SONG, X.D., ZHANG, Z., WANG, Y.W., YE, S.R. & HUANG, C.G. 2019 Reconstruction of RANS model and cross-validation of flow field based on tensor basis neural network. In *Proceedings of the ASME-JSME-KSME. 2019 8th Joint Fluids Engineering Conference*, pp. 1–6. Available at: https://doi.org/10.1115/AJKFluids2019-5572.
SPALART, P. 2023 An old-fashioned framework for machine learning in turbulence modeling. *ERCOFTAC Bull.* **134**. https://doi.org/10.48550/arXiv.2308.00837
VINUESA, R., NOORANI, A., LOZANO-DURÁN, A., KHOURY, G.K.E., SCHLATTER, P., FISCHER, P.F. & NAGIB, H.M. 2014 Aspect ratio effects in turbulent duct flows studied through direct numerical simulation. *J. Turbul.* **15** (10), 677–706.
VINUESA, R., PRUS, C., SCHLATTER, P. & NAGIB, H.M. 2016 Convergence of numerical simulations of turbulent wall-bounded flows and mean cross-flow structure of rectangular ducts. *Meccanica* **51** (12), 3025–3042.
VINUESA, R., SCHLATTER, P. & NAGIB, H.M. 2015 On minimum aspect ratio for duct flow facilities and the role of side walls in generating secondary flows. *J. Turbul.* **16** (6), 588–606.
VINUESA, R., SCHLATTER, P. & NAGIB, H.M. 2018 Secondary flow in turbulent ducts with increasing aspect ratio. *Phys. Rev. Fluids* **3**, 054606.
WILCOX, D.C. 1988 Reassessment of the scale-determining equation for advanced turbulence models. *AIAA J.* **26** (11), 1299–1310.
WITHERDEN, F.D. & JAMESON, A. 2017 Future directions of computational fluid dynamics. In *23rd AIAA Computational Fluid Dynamics Conference, 2017*, pp. 1–16. https://doi.org/10.2514/6.2017-3791
WU, J., XIAO, H., SUN, R. & WANG, Q. 2019*a* Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned. *J. Fluid Mech.* **869**, 553–586.
WU, J.-L., SUN, R., LAIZET, S. & XIAO, H. 2019*b* Representation of stress tensor perturbations with application in machine-learning-assisted turbulence modeling. *Comput. Meth. Appl. Mech. Engng* **346**, 707–726.
WU, J.-L., XIAO, H. & PATERSON, E. 2018 Physics-informed machine learning approach for augmenting turbulence models: a comprehensive framework. *Phys. Rev. Fluids* **7** (3), 1–28.

XIAO, H., WU, J.-L., LAIZET, S. & DUAN, L. 2020 Flows over periodic hills of parameterized geometries: a dataset for data-driven turbulence modeling from direct simulations. *Comput. Fluids* **200**, 104431.

ZHANG, Z., SONG, X.-D., YE, S.-R., WANG, Y.-W., HUANG, C.-G., AN, Y.-R. & CHEN, Y.-S. 2019 Application of deep learning method to Reynolds stress models of channel flow based on reduced-order modeling of DNS data. *J. Hydrodyn.* **31** (1), 58–65.