Computers and Fluids 202 (2020) 104497

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Data-driven modelling of the Reynolds stress tensor using random forests with invariance

Mikael L.A. Kaandorp[a,b,*], Richard P. Dwight[a]

[a] Aerodynamics, Faculty of Aerospace, Delft University of Technology, 2629 HS, Delft, the Netherlands
[b] Institute for Marine and Atmospheric research Utrecht (IMAU), Utrecht University, 3584 CC, Utrecht, the Netherlands

ARTICLE INFO

Article history:
Received 9 August 2019
Revised 31 December 2019
Accepted 4 March 2020
Available online 5 March 2020

Keywords:
Turbulence modelling
Machine-learning
Random forests
Reynolds anisotropy tensor
Non-linear eddy-viscosity closures

ABSTRACT

A novel machine learning algorithm is presented, serving as a data-driven turbulence modeling tool for Reynolds Averaged Navier-Stokes (RANS) simulations. This machine learning algorithm, called the Tensor Basis Random Forest (TBRF), is used to predict the Reynolds-stress anisotropy tensor, while guaranteeing Galilean invariance by making use of a tensor basis. By modifying a random forest algorithm to accept such a tensor basis, a robust, easy to implement, and easy to train algorithm is created. The algorithm is trained on several flow cases using DNS/LES data, and used to predict the Reynolds stress anisotropy tensor for new, unseen flows. The resulting predictions of turbulence anisotropy are used as a turbulence model within a custom RANS solver. Stabilization of this solver is necessary, and is achieved by a continuation method and a modified k-equation. Results are compared to the neural network approach of Ling et al. [29]. Results show that the TBRF algorithm is able to accurately predict the anisotropy tensor for various flow cases, with realizable predictions close to the DNS/LES reference data. Corresponding mean flows for a square duct flow case and a backward facing step flow case show good agreement with DNS and experimental data-sets. Overall, these results are seen as a next step towards improved data-driven modelling of turbulence. This creates an opportunity to generate custom turbulence closures for specific classes of flows, limited only by the availability of LES/DNS data.

1. Introduction

The last few years have seen the introduction of supervised machine learning (ML) algorithms as tools to exploit data for the purpose of modeling turbulence. RANS models are currently, and are expected to remain the norm for simulating turbulent flows in most industrial applications [46], because of their computational tractability, but suffer from poor accuracy and predictive power in a variety of important flows [9]. While a variety of nonlinear eddy-viscosity models (NLEVMs) and Reynolds-stress transport (RST) models have been developed using traditional techniques, it is the simplest linear eddy viscosity models such as the $k-\epsilon$ model and $k-\omega$ models that remain by far the most widely used. This has motivated some to advocate alternative modelling approaches that utilize available reference data more fully, and rely less on physical reasoning [10]. Supervised machine-learning methods developed in other fields, are – in the best case – able to distill large data-sets into simple functional relationships. This offers the hope of substantially improving current RANS models, by building closures customized for a particular class of flows based on appropriate reference LES or DNS data-sets [29,51]. However there exist significant obstacles to realizing these models in practice.

Firstly, a high sensitivity of the mean-flow to the detailed character of the turbulence has been reported – even in the case of channel flows [49]. This places stringent accuracy requirements on any data-derived closure model. Secondly, there exists no unique map from local mean-flow quantities to the needed turbulence statistics, due to non-local and non-equilibrium physics. Thirdly, any closure model should incorporate basic physical constraints, such as Galilean invariance: readily achievable in analytically derived models, but difficult when employing ML procedures which generate arbitrary functions. Fourthly, a ML model should produce smooth fields (they must be incorporated into a PDE solver), yet able to capture flows with rapid spatial variations, as in e.g. boundary-layers. Finally, RANS solvers are notoriously unstable and difficult to drive to convergence, even with standard closure models. Stabilization of a data-derived model is therefore necessary. These challenges are in addition to the standard ML challenges.

* Corresponding author at: Institute for Marine and Atmospheric research Utrecht (IMAU), Utrecht University, 3584 CC, Utrecht, the Netherlands.
E-mail address: m.l.a.kaandorp@uu.nl (M.L.A. Kaandorp).

https://doi.org/10.1016/j.compfluid.2020.104497

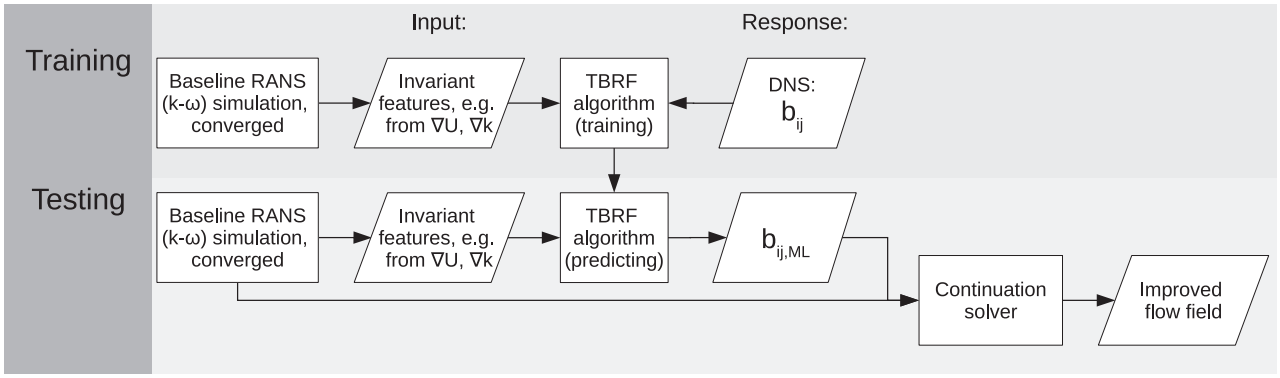
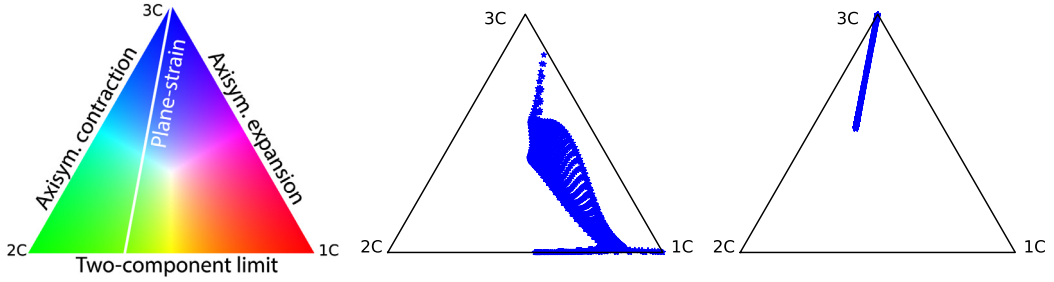
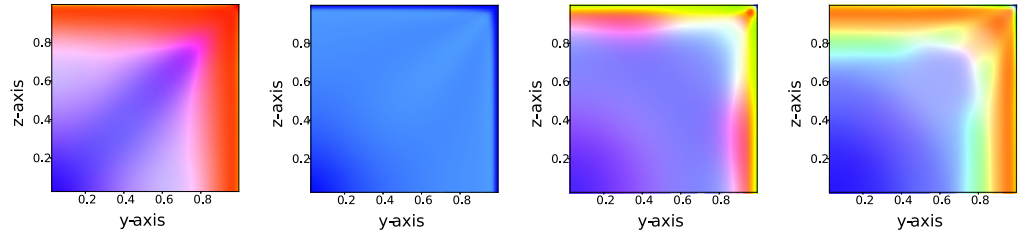


Fig. 1. Flow chart of the machine learning framework.



(a) Barycentric map (b) DNS Pinelli et al. (2010) (c) RANS $k-\omega$

Fig. 2. Barycentric map (transformation of Lumley triangle), for a turbulent square duct at Re = 3500. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) DNS Pinelli et al. (2010) (b) RANS $k-\omega$ (c) TBNN (d) TBRF

Fig. 3. Turbulence anisotropy state in the square duct (upper-right quadrant), visualized with the RGB colormap (Fig. 2(a)). TBNN = Tensor-Basis Neural-Network; TBRF = Tensor-Basis Random-Forest. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Think for example training against large volumes of data with high dimensionality [6], and avoiding overfitting, which can be done by using for example cross-validation methods, or methods specific to the algorithm used, e.g. randomly dropping connections in neural networks [48], or simplifying decision trees by pruning them back [14].

In this paper, a new approach is presented that addresses all these challenges to some extent, resulting in a closure model that significantly outperforms a baseline RANS model for a specified class of flows. The model is closely related to nonlinear eddy-viscosity models (NLEVMs), of e.g. [39], in which the normalized Reynolds-stress anisotropy tensor is predicted from the local mean-flow. We integrate a tensor basis for the anisotropy into a modified random-forest method, resulting in the Tensor Basis Random Forest (TBRF), analogously to the Tensor Basis Neural Network (TBNN) of Ling et al. [29]. Galilean invariance can therefore be guaranteed; and in comparison to artificial neural networks, our random forests are easier to implement, easier to train, have fewer hyperparameters, and have natural techniques for avoiding overfitting [16]. We introduce a method for stabilizing the RANS solver, using a combination of a modified k-equation, and relaxation against a Boussinesq stress-tensor. With this solver we can converge mean-flows to a steady state, with our TBRF closure model.

Many types of approach can be identified in the literature for improvements of RANS closure models with data, we only give a selection here. Broadly speaking there are parametric approaches, which calibrate or slightly modify existing models as in [7,12] (often with statistical inference); and structural approaches, which attempt to relax fundamental modelling assumptions, especially Boussinesq as in [29,52]. In the latter case, uncertainty quantification has been used to develop predictive models in the absence of data, by incorporating stochastic terms intended to capture the effect of modelling assumptions, see [13,50,57].

With data available, machine-learning has been used to capture potentially complex relationships between mean-flow and mod-

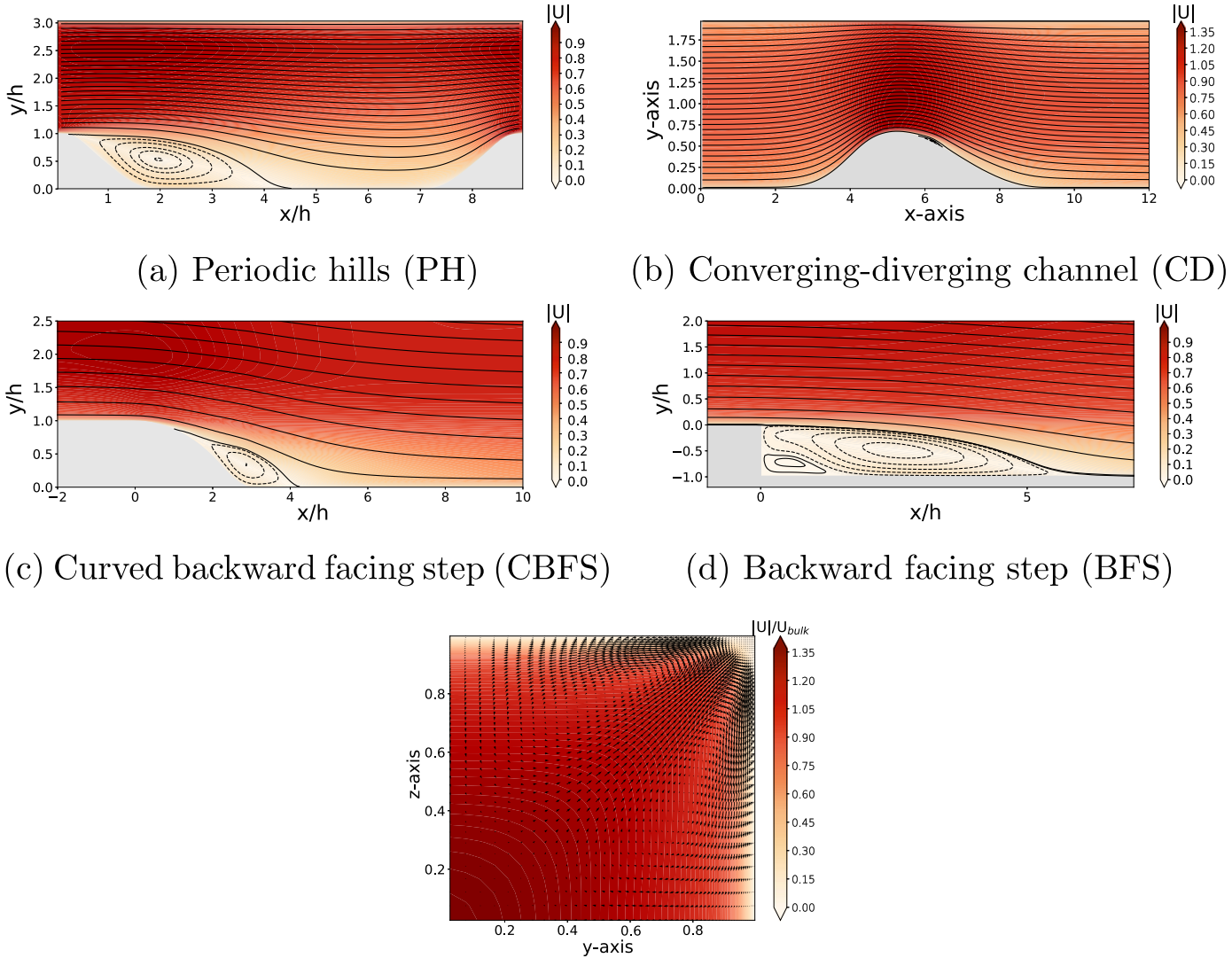


Fig. 4. Flow cases used in this work; the mean flow from the reference DNS/LES solutions is shown here. The color represents the velocity magnitude, and streamlines are plotted. Clockwise rotating regions of separation are indicated by dashed lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

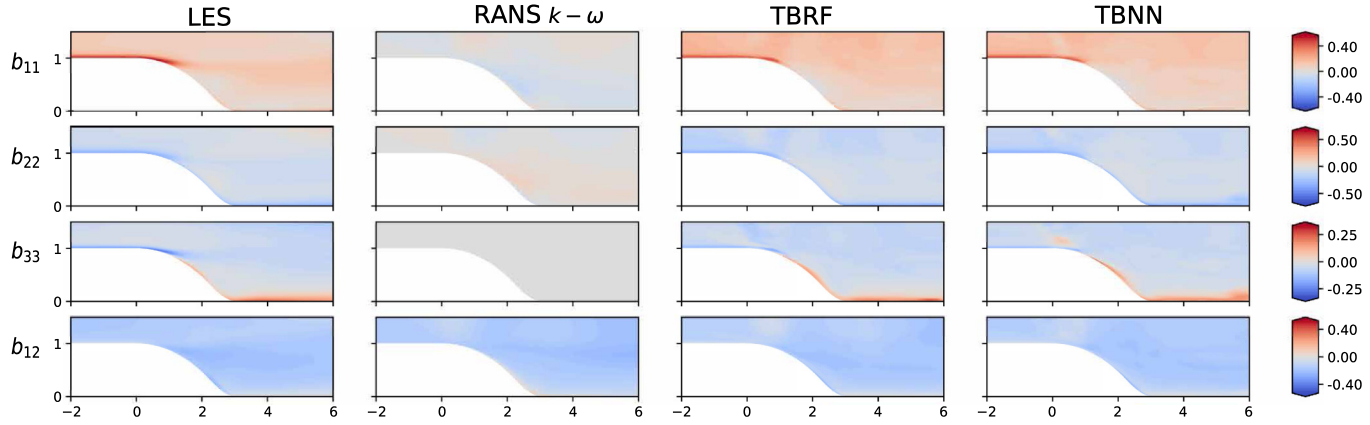


Fig. 5. Curved backward-facing step, $[\boldsymbol{b}]_{ij}$ components from LES, RANS, TBRF, and TBNN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

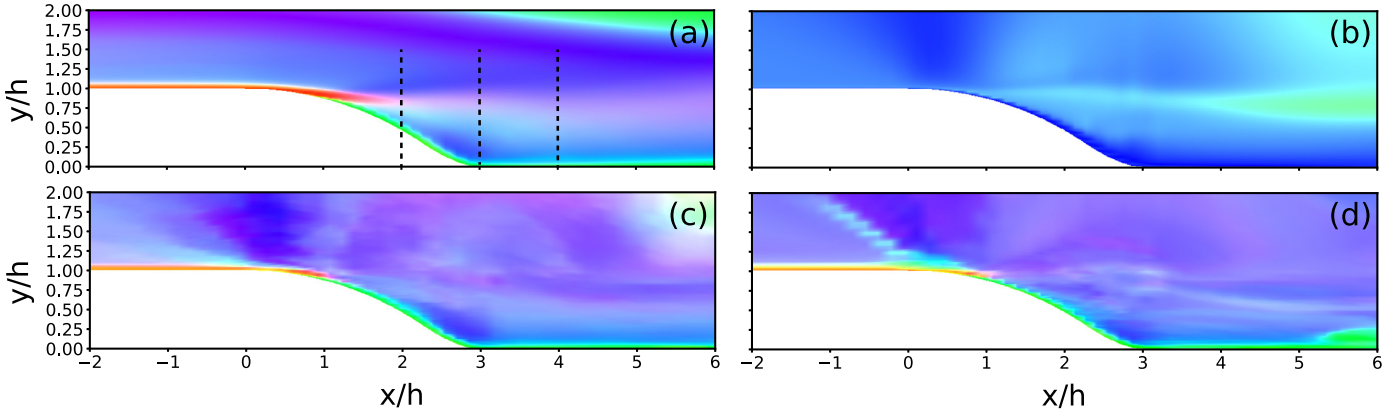


Fig. 6. Stress type for the curved backward-facing step, visualized with the RGB colormap (Fig. 2(a)): (a) LES from Bentaleb et al. [3]; (b) RANS $k-\omega$, (c) TBRF, and (d) TBNN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

elling terms. These approaches can largely be divided into those which solve inverse problems to identify local correction terms needed to match data, as in [11,36,45]; and those which apply machine-learning to directly predict model terms based on local mean-flow only, see [30]. The machine-learning methods used are various, Ling et al. [29] use neural networks; Ling and Co-authors [28,51] use random forests; and Weatheritt and Sandberg [53] uses gene-expression programming to obtain a concise form of their model. Despite the popularity of random-forests, existing authors have not incorporated frame-invariance, or solver stabilization that we introduce here. These developments are critical for the robustness and practicality of the method.

This paper is structured as follows. Section 2 will discuss the methodology for the TBRF framework, which includes underlying theory, the implementation of the algorithm itself, and the data-sets used in the framework. Section 3 will discuss the results from the framework, which include predictions for the anisotropy tensor, flow fields obtained after propagating these predictions into the flow field, and some proposals for future work with regards to quantifying uncertainty of the model output. Section 4 will present the conclusions.

2. Methods

When performing Reynolds-averaging of the incompressible Navier-Stokes equations, the complete effect of the unresolved turbulent fluctuations on the mean-flow is contained in a single term, $\nabla \cdot \boldsymbol{\tau}$ (where $[\boldsymbol{\tau}]_{ij} \coloneqq \overline{u_i' u_j'}$ is the Reynolds-stress tensor), which modifies the Navier-Stokes momentum equation [42]. To obtain a turbulence closure in this setting, it is sufficient to specify $\boldsymbol{\tau}$ in terms of mean-flow quantities $\bar{u}$ etc. The methodological goal of this work is to use plentiful DNS/LES data to estimate a nonlinear eddy-viscosity model (NLEVM) of the general form

$$\boldsymbol{\tau} \simeq \boldsymbol{h}(\nabla \bar{\boldsymbol{u}}, \dots),$$

where $\boldsymbol{h}(\cdot)$ is a function of mean-flow quantities only. This problem can be cast as a standard supervised machine learning task [35]. However we demand additionally that $\boldsymbol{h}$ is invariant to the choice of Galilean coordinate frame, and that the training process is computationally cheap and robust. The first is achieved with an integrity basis (Section 2.3), and the second by using a modified random forest (Section 2.5). The model for $\boldsymbol{\tau}$ thus obtained is used within a RANS solver to predict the mean-flow; this procedure requires solver stabilization (described in Section 2.7). Finally, training and validation cases with DNS/LES data are listed in Section 3.1.

2.1. Outline of framework for data-driven turbulence modeling

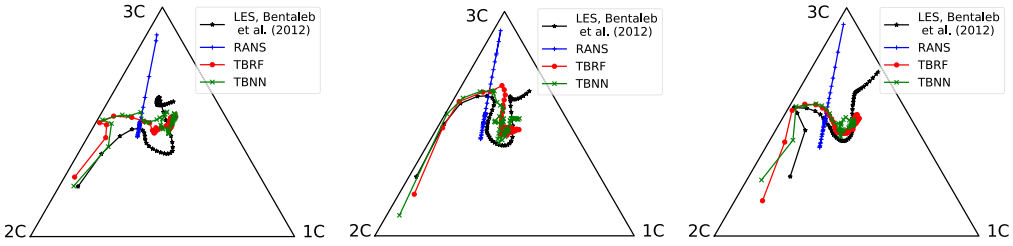
Our framework consists of a training phase and a prediction phase, see Fig. 1. In the training phase, high-fidelity DNS/LES data is collected for a number of test-cases. The data should ideally contain full first- and second-order single-point statistics highly resolved in the spatial domain, from which the normalized anisotropy tensor, $\boldsymbol{b}$, is computed, see Section 2.2. These same test-cases are simulated with RANS using the $k-\omega$ closure model, and input features are derived from the mean-flow solution at every node of the spatial mesh. The machine learning algorithm is then trained to approximate the DNS ground-truth anisotropy tensor, from the RANS inputs over the full solution fields of all training-cases simultaneously.

As we use a greedy algorithm to train the decision-trees, the training cost for a data-set of size N is $\mathcal{O}(N \log^2 N)$, so there is no practical restriction on the number of cases which can be used in training (indeed a much more severe limitation has proven to be the availability of high-quality DNS/LES data). However, we do not expect the map from the mean-flow to unresolved turbulence statistics to be unique, even for a very limited class of flows. So, in order to capture this non-uniqueness and to prevent overfitting, multiple decision-trees are trained across random subsets of the data.

In the prediction phase, for a previously unseen flow case, the anisotropy tensor is estimated using the mean of the random forest predictions, with input from a baseline RANS mean-field. An updated mean-field is obtained by solving a modified RANS system with the momentum equation supplemented by the predicted anisotropy.

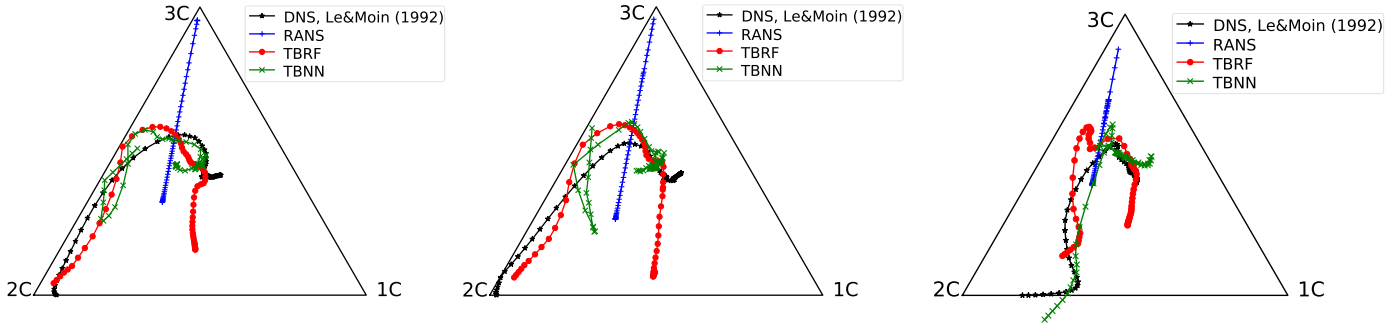
We want to stress that this is a corrective approach, with a single ML prediction providing an updated solution, similar to that practiced in [28,55]. In other words, it is not a replacement for a 'standard' turbulence model, where the Reynolds stress is assumed to be a specific function of the mean flow properties, which is then iteratively solved until convergence is achieved. Such an iterative approach would in theory also be conceivable, in which ML predictions of $\boldsymbol{b}$ and modified RANS form a closed loop. In this case the ML should be trained on the DNS mean-field, not the RANS field. However such ambitious approaches are currently untested, it is unclear under what conditions the coupled system will converge, and whether the converged solution will resemble ground-truth. This is however an important topic for further research.

Adding to this point, it should be stressed that the approach here is only tested for steady RANS flow cases. It is untested for

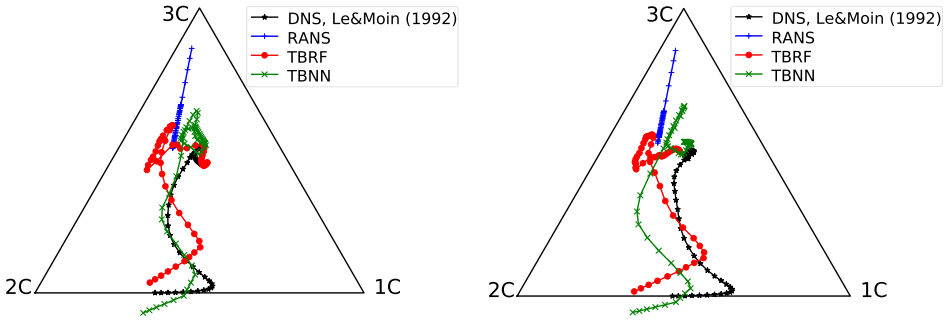


(a) $x/h = 2.0$ (b) $x/h = 3.0$ (c) $x/h = 4.0$

Fig. 7. Barycentric map data at three sections (see Fig. 6a) for the curved backward-facing step. Comparing LES [3], RANS $k-\omega$ simulation, TBRF, and TBNN.



(a) $x/h = 4.0$ (b) $x/h = 6.0$ (c) $x/h = 10.0$



(d) $x/h = 15.0$ (e) $x/h = 19.0$

Fig. 8. Barycentric map data at five sections for the backward-facing step. Comparing LES [25], RANS $k-\omega$, TBRF, and TBNN.

unsteady flows such as present in low-pressure turbines, see e.g. [1].

The ground truth for $\boldsymbol{b}$ coming from high-quality DNS/LES data will still contain some error due to e.g. dicretization errors, errors due to a finite averaging time, and possibly model errors. The errors in the ground truth are assumed to be small here (compared to the model errors of the machine learning algorithms), and therefore not taken in account. Supervised machine learning methods exist which theoretically could be able to capture uncertainties in the ground truth (see e.g. Gaussian process regression [41])

2.2. Reynolds stress and realizability constraints

First we briefly review some basic properties of the Reynolds stresses, relevant for constructing a meaningful ML model. Firstly $\boldsymbol{\tau}$ can be intrinsically divided into isotropic and anisotropic (deviatoric) parts:

$$\boldsymbol{\tau} = \frac{2}{3}k\boldsymbol{I} + \boldsymbol{a}, \tag{1}$$

where $\boldsymbol{a}$ is the Reynolds stress anisotropy tensor, $k := \frac{1}{2}\text{trace}(\boldsymbol{\tau})$ is the turbulent kinetic energy, and $\boldsymbol{I}$ is the identity. It is the anisotropic part of the Reynolds stresses which is important: only this part is effective in transporting momentum, while the isotropic stresses can be simply absorbed into a modified pressure term [40]. The *non-dimensional* Reynolds stress anisotropy tensor, $\boldsymbol{b}$, is defined as:

$$\boldsymbol{b} := \frac{\boldsymbol{\tau}}{2k} - \frac{1}{3}\boldsymbol{I}, \tag{2}$$

and this is the quantity which we attempt to predict with machine learning. In the remainder of this paper "anisotropy tensor" will refer to $\boldsymbol{b}$. To model $\boldsymbol{b}$, eddy-viscosity closure models typically make the intrinsic assumption that $\boldsymbol{b}$ is function of local mean-flow quantities only. Linear eddy-viscosity models such as $k-\epsilon$ and $k-\omega$ [23,54], go on to make the specific, Boussinesq assumption that $\boldsymbol{b} \simeq \frac{1}{2}\nu_t(\nabla\bar{\boldsymbol{u}} + \nabla\bar{\boldsymbol{u}}^T) = \nu_t\hat{\boldsymbol{S}}$ for some scalar *eddy viscosity* $\nu_t(\boldsymbol{x})$, which remains to be specified. Both the intrinsic and specific assumptions introduce modelling error. We aim to estimate

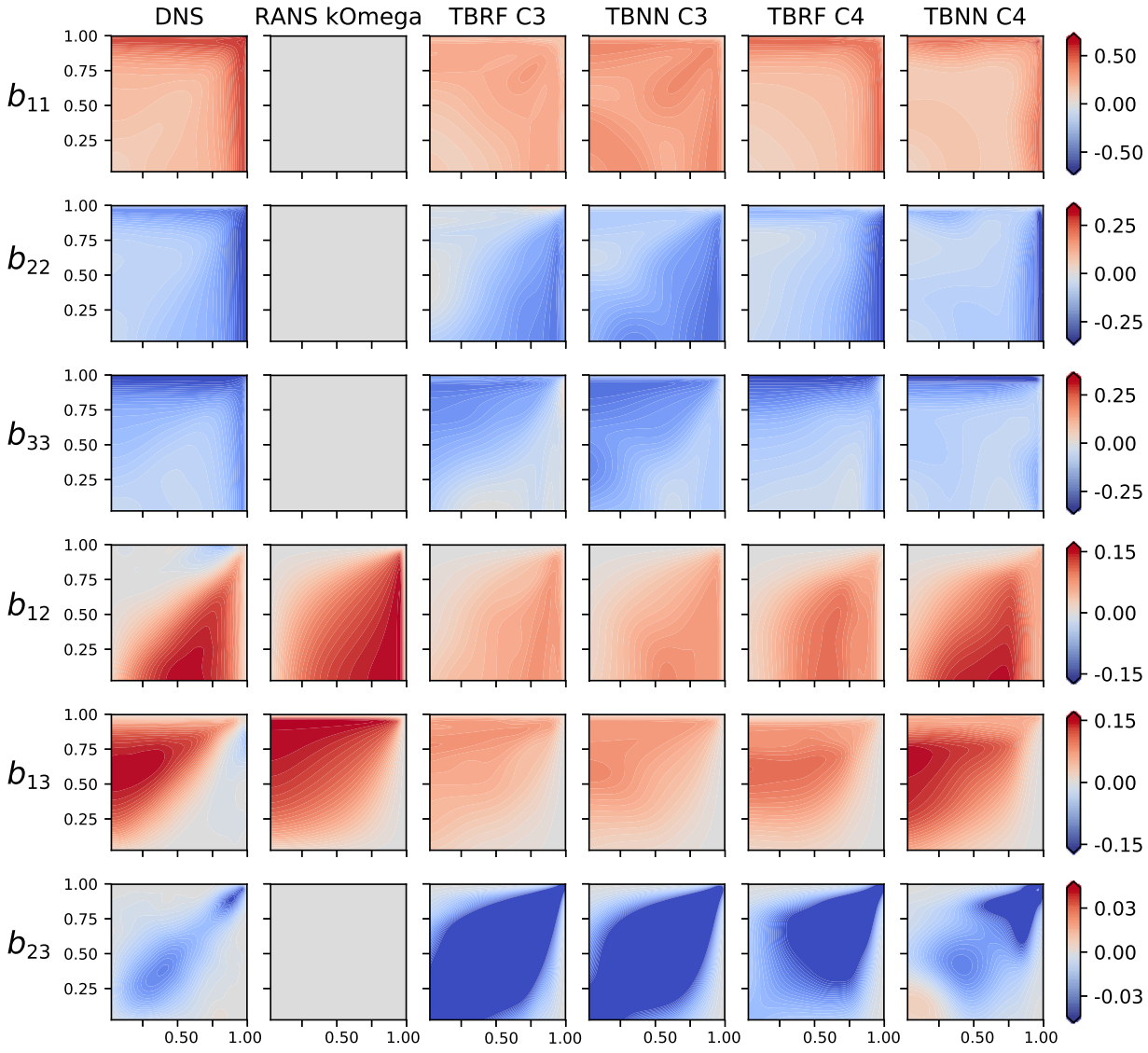


Fig. 9. Square duct, $[\boldsymbol{b}]_{ij}$ components from DNS, RANS, TBRF, and TBNN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and reduce the error in the latter with LES databases and machine-learning.

The properties of $\boldsymbol{\tau}$ and $\boldsymbol{b}$ lead to physical constraints on models and means of visualization. A matrix $\boldsymbol{A}$ is positive semi-definite if (and only if) $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq 0$, $\forall \boldsymbol{x} \in \mathbb{R}^N$. Since the outer product of any vector $\boldsymbol{u}'$ with itself ($\boldsymbol{u}'\otimes\boldsymbol{u}'$) is positive semi-definite; and since the Reynolds stress is an arithmetic average of such tensors, it is also positive semi-definite. As trivial consequences, all eigenvalues of $\boldsymbol{\tau}$ are real and positive, and

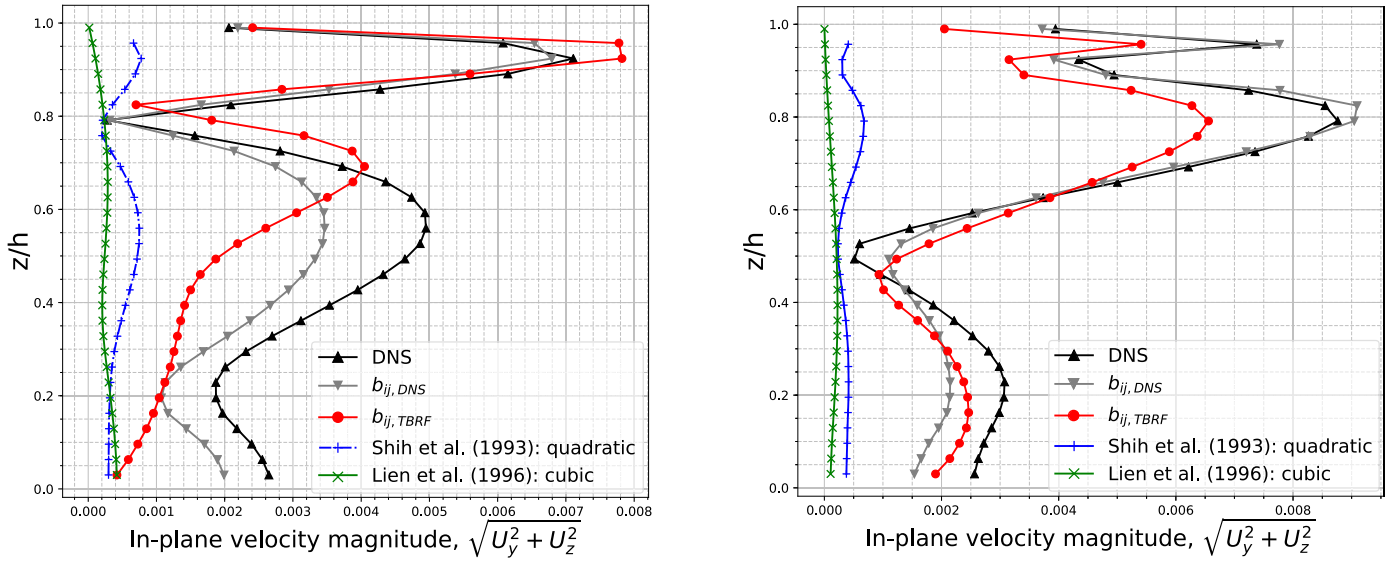
$$\tau_{\alpha\alpha} \geq 0 \quad \forall \alpha \in \{1,2,3\}, \quad \det(\boldsymbol{\tau}) \geq 0, \quad \tau_{\alpha\beta}^2 \leq \tau_{\alpha\alpha}\tau_{\beta\beta} \quad \forall \alpha \neq \beta. \tag{3}$$

These properties of $\boldsymbol{\tau}$ have implications for $\boldsymbol{b}$. Let the eigenvalues of $\boldsymbol{\tau}$ be ϕ_i, and those of $\boldsymbol{b}$ be λ_i, then

$$\lambda_i = \frac{\phi_i}{2k} - \frac{1}{3}, \tag{4}$$

and both the eigenvalues and diagonal components of $\boldsymbol{b}$ are in the interval $[-\frac{1}{3}, \frac{2}{3}]$. Furthermore using the Cauchy-Schwarz inequality in (3) the off-diagonal components of $\boldsymbol{b}$ are in $[-\frac{1}{2}, \frac{1}{2}]$. Since $\mathrm{trace}(\boldsymbol{b}) = 0$ only two independent invariants of the anisotropy tensor exist, e.g.: $\mathrm{II} := [\boldsymbol{b}]_{ij}[\boldsymbol{b}]_{ji}$ and $\mathrm{III} := [\boldsymbol{b}]_{ij}[\boldsymbol{b}]_{in}[\boldsymbol{b}]_{jn}$. Therefore in the II-III plane all realizable states of turbulence anisotropy can be plotted, which are further restricted to a triangular domain corresponding to the constraints on $\boldsymbol{b}$ just mentioned. This leads to the well-known "Lumley triangle" of Lumley and Co-authors [31,32] which captures the anisotropic state of turbulence. The lesser known barycentric map was introduced in [2], and is a transformation of the Lumley triangle into barycentric coordinates, and will be used for the purposes of visualization and comparison in this paper, see Fig. 2.

These triangles highlight three limiting states of turbulence anisotropy: 1-component turbulence (restricted to a line, one eigenvalue of $\boldsymbol{b}$ is non-zero), 2-component turbulence (restricted to a plane, two eigenvalues of $\boldsymbol{b}$ are non-zero), and 3-component turbulence (isotropic turbulence, three eigenvalues are non-zero). Fig. 2 shows these, along with invariants for a square-duct flow simulated with DNS from Pinelli et al. [38], and a $k-\omega$ RANS simulation. For any 2d linear eddy-viscosity RANS simulation, the predicted anisotropy invariants will lie entirely along the line



(a) $y/h = 0.5$ (b) $y/h = 0.8$

Fig. 10. In-plane mean velocity profiles at two sections of the square duct. Comparing: DNS, and the mean velocity fields obtained by propagating the DNS anistropy, and the TBRF-predicted anisotropy (labeled $b_{ij,DNS}$ and $b_{ij,TBRF}$ respectively). Also shown are two non-linear eddy-viscosity models.

indicated as "plane strain". This illustrates the inability of linear eddy-viscosity models to adequately represent anisotropy.

One further method of visualization we will use is the Red-Green-Blue (RGB) map, in which each anisotropic state is assigned a color and the flow domain is colored accordingly, in a kind of generalized contour plot [50]. Fig. 2(a) presents this colormap, and in Fig. 3 the colormap is applied to the square-duct with DNS and RANS data (the same data used for Fig. 2(b-c)). The DNS data shows 1-component turbulence close to the wall, and 3-component near the centreline of the duct, whereas the RANS simulation only represents turbulence near the 3-component limit. Also in this figure (c) and (d), machine-learning predictions of the same invarients are plotted, to be discussed later in Section 3.

2.3. Invariance of tensor-valued functions

The Navier-Stokes equations are Galilean invariant, i.e. unchanged by choice of inertial frame. It is a physical requirement that any model for the anisotropy tensor also be frame invariant, and thereby satisfy the simple requirement that the functional model should not depend on the choice of coordinate system. In fact this proves to be a critical requirement for the success of our machine-learning strategy, see Section 3. Let $\boldsymbol{Q} \in \mathbb{R}^{3\times3}$ be an arbitrary real orthogonal transformation; then a scalar-valued function $f : \mathbb{R}^{3\times3} \to \mathbb{R}$, with tensor argument $\boldsymbol{S} \in \mathbb{R}^{3\times3}$ is frame invariant if and only if:

$$f(\boldsymbol{S}) = f(\boldsymbol{Q}\boldsymbol{S}\boldsymbol{Q}^T), \quad \forall \boldsymbol{Q}, \boldsymbol{S}. \tag{5}$$

Similarly a tensor-valued function $\boldsymbol{h} : \mathbb{R}^{3\times3} \to \mathbb{R}^{3\times3}$ is frame invariant if and only if (e.g. [47]):

$$\boldsymbol{Q}\,\boldsymbol{h}(\boldsymbol{S})\boldsymbol{Q}^T = \boldsymbol{h}(\boldsymbol{Q}\boldsymbol{S}\boldsymbol{Q}^T) \quad \forall \boldsymbol{Q}, \boldsymbol{S}. \tag{6}$$

One means of finding a $\boldsymbol{h}$ satisfying (6), is to start with a scalar function $h : \mathbb{R} \to \mathbb{R}$, and specify that $\boldsymbol{h}$ is a natural generalization of h. See Higham [17] for several standard definitions for $\boldsymbol{h}$ given h (e.g. in terms of power-series). Under all these definitions, not only do we have $\boldsymbol{h}(\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^{-1}) = \boldsymbol{X}\boldsymbol{h}(\boldsymbol{S})\boldsymbol{X}^{-1}$ for any invertible matrix $\boldsymbol{X}$ (implying frame invariance by setting $\boldsymbol{X} \equiv \boldsymbol{Q}$); but also other properties such as $\boldsymbol{h}(\boldsymbol{S}^T) = \boldsymbol{h}(\boldsymbol{S})^T$ and $\boldsymbol{\lambda} = \text{eigval}\{\boldsymbol{S}\} \Rightarrow h(\boldsymbol{\lambda}) = \text{eigval}\{\boldsymbol{h}(\boldsymbol{S})\}$.

In addition we assume that $\boldsymbol{h}$ has a power-series representation

$$\boldsymbol{h}(\boldsymbol{S}) = \sum_{i=0}^{\infty} a^{(i)}(\boldsymbol{\lambda})\boldsymbol{S}^i, \quad \boldsymbol{\lambda} = \text{eigval}\{\boldsymbol{S}\}, \quad a^{(i)} : \mathbb{R}^3 \to \mathbb{R},$$

for some scalar-valued functions $a^{(i)}$ whose arguments are the invariants of $\boldsymbol{S}$. We reduce this infinite sum to a finite sum with the following trick: by the Cayley–Hamilton theorem, every matrix satisfies its own characteristic equation $q(\boldsymbol{S}) = \boldsymbol{0}$. However q is a polynomial of degree 3 (in 3d), whose coefficients are functions only of the invariants of $\boldsymbol{S}$. Hence using q we can recurrsively express powers $\boldsymbol{S}^3$ and higher in terms of $\boldsymbol{I}$, $\boldsymbol{S}$ and $\boldsymbol{S}^2$. As a result there must exist an equivalent expression for $\boldsymbol{h}$:

$$\boldsymbol{h}(\boldsymbol{S}) = \sum_{i=0}^{2} \tilde{a}^{(i)}(\boldsymbol{\lambda})\boldsymbol{S}^i,$$

for some different scalar-valued functions $\tilde{a}^{(i)} : \mathbb{R}^3 \to \mathbb{R}$.

In our application we seek a function from *multiple* tensors to the anisotropy tensor $\boldsymbol{b}$, meaning a generalization of the above is required. The result remains – under the above assumptions – that the most general $\boldsymbol{h}$ can be written in terms of a finite tensor-basis known as the *integrity basis*.

In particular when deriving nonlinear eddy-viscosity models, it is sometimes assumed that the Reynolds stresses are a function of the local, normalized, mean rates of strain $\boldsymbol{S}$ and rotation $\boldsymbol{R}$. I.e. $\boldsymbol{b} := \boldsymbol{b}(\boldsymbol{S}, \boldsymbol{R})$, where

$$\boldsymbol{S} = \frac{1}{2}\frac{k}{\epsilon}\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right), \quad \boldsymbol{R} = \frac{1}{2}\frac{k}{\epsilon}\left(\nabla\boldsymbol{u} - \nabla\boldsymbol{u}^T\right). \tag{7}$$

In this case [39] there are 10 integrity basis tensors $\boldsymbol{T}^{(m)}$, making the most general expression for $\boldsymbol{b}$:

$$\boldsymbol{b} = \boldsymbol{h}(\boldsymbol{S}, \boldsymbol{R}) = \sum_{m=1}^{10} \boldsymbol{T}^{(m)}(\boldsymbol{S}, \boldsymbol{R})\, g^{(m)}(\theta_1, \ldots, \theta_5), \tag{8}$$

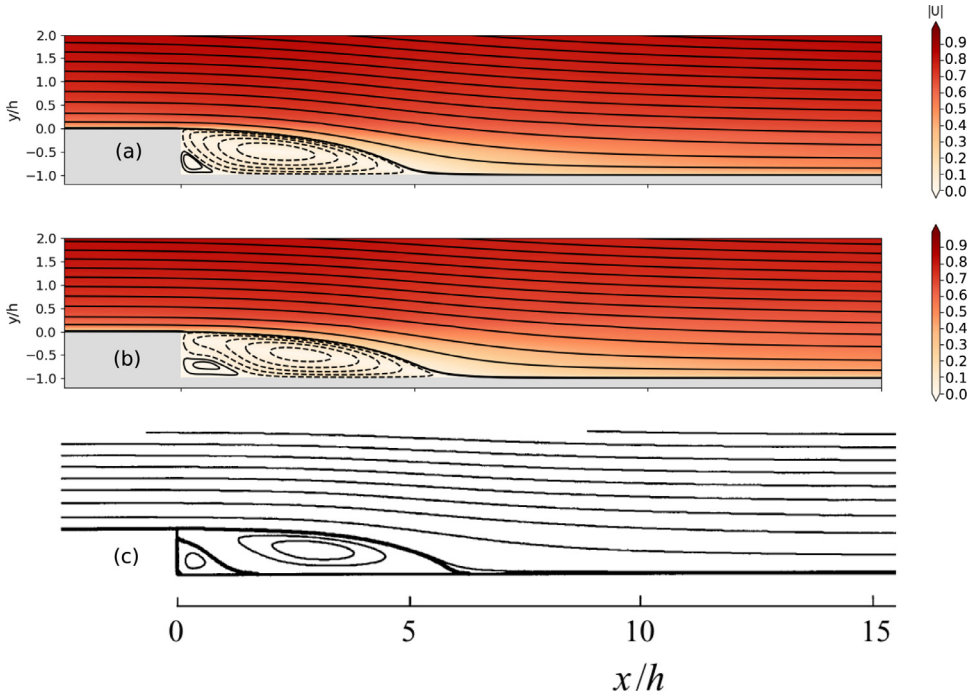


Fig. 11. Comparison of the streamlines for the backward facing step as given by (*a*) the RANS $k-\omega$ simulation, (*b*) the propagated flow field using $b_{ij,\mathrm{TBRF}}$, and (*c*) DNS, adapted from Le et al. [26]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

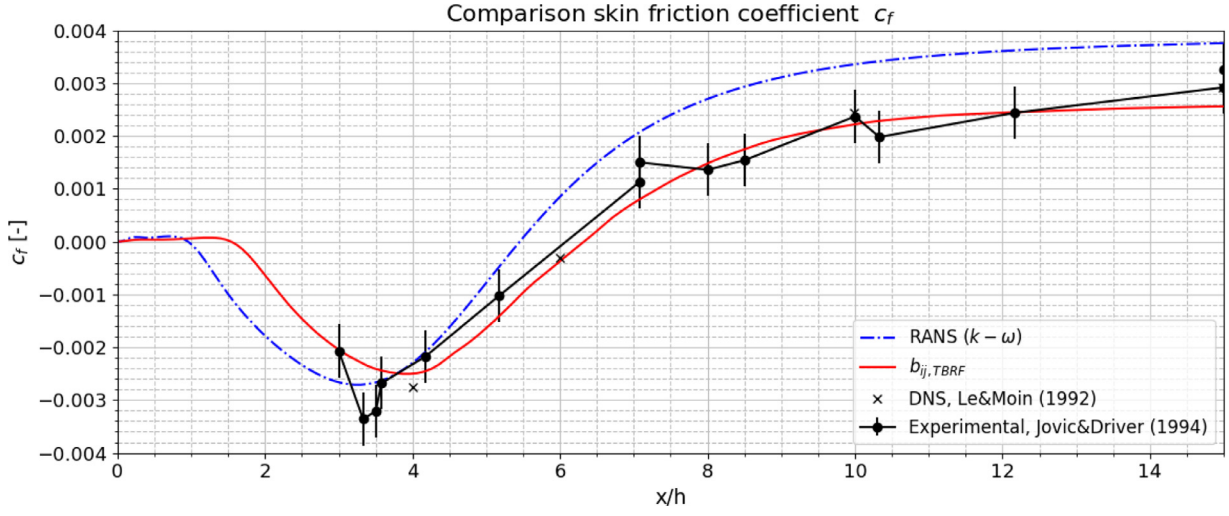


Fig. 12. Skin-friction coefficient for the backward facing step. Experimental data from Jovic and Driver [18]; DNS data from Le and Moin [25].

where $g^{(m)}$ are scalar functions of the invariants θ_i. The basis tensors derived from $\boldsymbol{S}$ and $\boldsymbol{R}$ are [39]:

$$\boldsymbol{T}^{(1)} = \boldsymbol{S} \qquad \boldsymbol{T}^{(6)} = \boldsymbol{R}^2\boldsymbol{S} + \boldsymbol{S}\boldsymbol{R}^2 - \tfrac{2}{3}\boldsymbol{I}\cdot\mathrm{trace}(\boldsymbol{S}\boldsymbol{R}^2)$$
$$\boldsymbol{T}^{(2)} = \boldsymbol{S}\boldsymbol{R} - \boldsymbol{R}\boldsymbol{S} \qquad \boldsymbol{T}^{(7)} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{R}^2 - \boldsymbol{R}^2\boldsymbol{S}\boldsymbol{R}$$
$$\boldsymbol{T}^{(3)} = \boldsymbol{S}^2 - \tfrac{1}{3}\boldsymbol{I}\cdot\mathrm{trace}(\boldsymbol{S}^2) \qquad \boldsymbol{T}^{(8)} = \boldsymbol{S}\boldsymbol{R}\boldsymbol{S}^2 - \boldsymbol{S}^2\boldsymbol{R}\boldsymbol{S}$$
$$\boldsymbol{T}^{(4)} = \boldsymbol{R}^2 - \tfrac{1}{3}\boldsymbol{I}\cdot\mathrm{trace}(\boldsymbol{R}^2) \qquad \boldsymbol{T}^{(9)} = \boldsymbol{R}^2\boldsymbol{S}^2 + \boldsymbol{S}^2\boldsymbol{R}^2 - \tfrac{2}{3}\boldsymbol{I}\cdot\mathrm{trace}(\boldsymbol{S}^2\boldsymbol{R}^2)$$
$$\boldsymbol{T}^{(5)} = \boldsymbol{R}\boldsymbol{S}^2 - \boldsymbol{S}^2\boldsymbol{R} \qquad \boldsymbol{T}^{(10)} = \boldsymbol{R}\boldsymbol{S}^2\boldsymbol{R}^2 - \boldsymbol{R}^2\boldsymbol{S}^2\boldsymbol{R}$$

with invariants

$$\theta_1 = \mathrm{trace}(\boldsymbol{S}^2) \quad \theta_2 = \mathrm{trace}(\boldsymbol{R}^2) \quad \theta_3 = \mathrm{trace}(\boldsymbol{S}^3)$$
$$\theta_4 = \mathrm{trace}(\boldsymbol{R}^2\boldsymbol{S}) \quad \theta_5 = \mathrm{trace}(\boldsymbol{R}^2\boldsymbol{S}^2).$$

In [51] this approach is extended to derive a set of 47 invariants based on $\nabla\tilde{\boldsymbol{u}}$, $\nabla\boldsymbol{k}$, and $\nabla\boldsymbol{p}$. This is the system we use in the following; the full feature-set will be shown in Section 2.6.

2.4. Tensor basis neural network (TBNN)

In [28] an artificial neural network was used to represent $\boldsymbol{h}$. By careful choice of the network topology the idea of the tensor basis is encoded into the network. The network contains a 5-node input layer receiving the 5 scalar invariants derived from $\boldsymbol{S}$ and $\boldsymbol{R}$. These inputs then go through a densely connected feed-forward network with 10 hidden layers. Additionally, the network contains an extra input layer consisting of the 10 base tensors $\boldsymbol{T}^{(m)}$. The output of the feed-forward network (representing the $g^{(m)}$ functions) is merged with this extra input layer, reproducing (8). Thereby the tensor-basis form of $\boldsymbol{h}$ and Galilean invariance is achieved.

In this work TBNN is used as a competing method for comparison. The implementation was obtained from the authors of [29]: it is written in python using the lasagne library for build-

ing and training the neural network (source code available at github.com/tbnn/tbnn). The same settings as there were used to aid a fair comparison. A leaky ReLU activation function was used. The number of hidden layers and nodes-per-layer were optimized in [29], and those values were used here. The TBNN is trained using the Adam algorithm [21], with the learning rate (2.5×10^{-5}), the learning-rate decay, and the mini-batch size (1000) again based on Ling et al. [29].

Neural networks in general are challenging to train, and this was no exception. To avoid overfitting, the data was randomly partitioned into training (80%) and validation (20%) sets. Early-stopping was used, which terminates training when the training error reduces, but the validation error starts consistently increasing. Since the validation error as a function of the epoch has a noisy behaviour, a moving average of five samples was taken to determine when early-stopping should be activated. Initial network weights were randomly chosen, and the TBNN was trained five times from which the network was selected which performed best on the validation set.

2.5. Tensor basis random forest (TBRF)

Decision trees base their predictions on a series of if-then tests on the input. Random forests consist of collections of decision trees with some randomized component differentiating trees. Multiple decision tree algorithms exist, of which the CART (Classification And Regression Tree) algorithm serves as the starting point for the Tensor Basis Decision Tree (TBDT), which is used in the Tensor Basis Random Forest (TBRF). A brief overview of the TBRF algorithm is presented here, for a more technical overview the reader is referred to the appendix.

In the training phase of the CART decision tree, the feature space is recursively split into two bins. In each bin a constant value is used to approximate the training data. Given p features let the training data consist of $\boldsymbol{X} \in \mathbb{R}^{p\times N}$ point locations in feature-space, and corresponding $\boldsymbol{y} \in \mathbb{R}^N$ scalar output values. Each split is aligned with an input feature, and therefore the location of the split is completely specified by a splitting feature index $j \in \{1, \dots, p\} \subset \mathbb{N}$, and value s. The two bins in which the data is split are denoted $R_L \subset \mathbb{R}$ (left) and $R_R \subset \mathbb{R}$ (right), and are given by

$$R_L(j,s) = \{\boldsymbol{X} \mid [\boldsymbol{X}]_j \leq s\} \quad R_R(j,s) = \{\boldsymbol{X} \mid [\boldsymbol{X}]_j > s\}. \tag{9}$$

For the TBDT, constant values are chosen for the tensor basis coefficients $g^{(m)}$ (see (8)) in each bin, which will be denoted by $g_L^{(m)}$ and $g_R^{(m)}$ for R_L and R_R respectively. The values are chosen such that the mismatch with respect to the reference DNS/LES anisotropy tensor $\boldsymbol{b}$ is minimized. The cost function can be defined as:

$$J = \sum_{x_i \in R_L(j,s)} \left\| \sum_{m=1}^{10} \boldsymbol{T}_i^{(m)} g_L^{(m)} - \boldsymbol{b}_i \right\|_F^2 + \sum_{x_i \in R_R(j,s)} \left\| \sum_{m=1}^{10} \boldsymbol{T}_i^{(m)} g_R^{(m)} - \boldsymbol{b}_i \right\|_F^2, \tag{10}$$

where the Frobenius norm is used to calculate the difference between the reference DNS/LES anisotropy tensor, and the tensor resulting from the tensor basis. It can be shown by setting the derivative of J with respect to $g^{(m)}$ in each bin to zero, the optimum value for the 10 tensor basis coefficients in each bin is found using

$$\boldsymbol{g} = \left(\sum_{i=1}^{N} \hat{\boldsymbol{T}}_i^T \hat{\boldsymbol{T}}_i \right)^{-1} \left(\sum_{i=1}^{N} \hat{\boldsymbol{T}}_i^T \hat{\boldsymbol{b}}_i \right). \tag{11}$$

were $\hat{\boldsymbol{T}}_i$ and $\hat{\boldsymbol{b}}_i$ are matrices containing the flattened basis tensors and reference DNS/LES anisotropy tensors:

$$\hat{\boldsymbol{T}}_i = \begin{bmatrix} [\boldsymbol{T}_i^{(1)}]_{11} & [\boldsymbol{T}_i^{(2)}]_{11} & \cdots & [\boldsymbol{T}_i^{(10)}]_{11} \\ [\boldsymbol{T}_i^{(1)}]_{12} & [\boldsymbol{T}_i^{(2)}]_{12} & \cdots & [\boldsymbol{T}_i^{(10)}]_{12} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{T}_i^{(1)}]_{33} & [\boldsymbol{T}_i^{(2)}]_{33} & \cdots & [\boldsymbol{T}_i^{(10)}]_{33} \end{bmatrix}, \quad \hat{\boldsymbol{b}}_i = \begin{bmatrix} [\boldsymbol{b}_i]_{11} \\ [\boldsymbol{b}_i]_{12} \\ \vdots \\ [\boldsymbol{b}_i]_{33} \end{bmatrix}. \tag{12}$$

In other words, during the training of the TBDT, each split in the tree is made by solving two least squares problems for $g^{(m)}$, for each j, and each value s, and selecting the combination which minimizes J. The outer minimization is solved by brute-force over features j, and one-dimensional optimization is used over s. The Brent 1d-optimization algorithm is used, offering a good trade-off between speed and robustness (by falling back on the golden section search in the worst case) [4]. When the number of samples in a bin falls below a threshold, we switch to brute-force search for s. This threshold was set to 150 samples to limit the training time of the decision trees, a sensitivity study showed no significant influence of the threshold on the performance of the TBRF algorithm. The splitting of the TBDT branches is terminated at either a specified maximum branching depth, or at a minimum number of samples per bin. Due to the redundancy of the tensor-basis for any given sample $i \in \{1, \dots, N\}$, (11) can become ill-posed, especially towards the leaves of the tree, when only a few samples remain in a bin. Therefore some L^2-regularization is added to J with coefficient $\Gamma \in \mathbb{R}^+$, c.f. ridge regression, see the appendix for more details.

In the tensor basis random forest, multiple tensor basis decision trees trained on a collection of bagged data sets are averaged. Bagging implies data is repeatedly randomly drawn (with replacement) from the full data-set. Bagging is expected to work well when combining a number of high-variance, low-bias estimators, such as the decision tree. By averaging many noisy, but unbiased models, the variance of the prediction is reduced [16]. The variance of the predictions will be reduced most effectively if the errors in the component models are as far as possible uncorrelated. This is encouraged by introducing some additional randomness in the individual trees: at each split, not all features, but a randomly selected subset of the available features is used for splitting. The specific parameters used in our computations will be stated in Section 3.

Instead of taking the mean over all the tensor basis decision trees, it proved to be more successful to take the median of the trees in the random forest (as for example investigated in [43]), since this removed sensitivity to outliers in the predictions, see the appendix for a comparison. Since the random forest is a piecewise constant approximation of $\boldsymbol{b}$, and derivatives of $\boldsymbol{b}$ are needed in the N-S equation, the predictions from the TBRF are smoothed spatially with a Gaussian filter, before they are propagated through the solver to obtain a flow field (see Section 2.7). The TBRF algorithm has no explicit spatial correlation in the predictions since these are based on local features of the flow, so filtering the predictions will introduce some spatial correlation. The filter standard deviation was set to 3 cell lengths, as this sufficiently smooths the predictions while maintaining all important features of the predicted tensor (see the appendix for more details). This filter width is an ad hoc choice, and can possibly be adjusted more specifically for numerical stability in future work by looking at e.g. required condition numbers for the solver (see e.g. [56]).

Several benefits can be identified in using the TBRF algorithm. First of all, the available data can be divided into a training data-set and validation data-set in a natural manner. Since random sam-

Table 1
Features used for the machine learning algorithms, obtained from Wang et al. [51] and Wu et al. [55]. For features with an * all cyclic permutations of labels of anti-symmetric tensors need to be taken in account. For FS1 and FS2 the trace of the tensor quantities is taken. Features marked with † are rotationally invariant but not Galilean invariant.

Set	Features	Normalization	Comment
FS1	$\boldsymbol{S}^2$, $\boldsymbol{S}^3$, $\boldsymbol{R}^2$, $\boldsymbol{R}^2\boldsymbol{S}$, $\boldsymbol{R}^2\boldsymbol{S}^2$, $\boldsymbol{R}^2\boldsymbol{S}\boldsymbol{R}\boldsymbol{S}^2$	–	Invariant set based on $\boldsymbol{S}$ and $\boldsymbol{R}$
FS2	$\boldsymbol{A_k}^2$, $\boldsymbol{A_k}^2\boldsymbol{S}$, $\boldsymbol{A_k}^2\boldsymbol{S}^2$, $\boldsymbol{A_k}^2\boldsymbol{S}\boldsymbol{A_k}\boldsymbol{S}^2$, $\boldsymbol{R}\boldsymbol{A_k}$, $\boldsymbol{R}\boldsymbol{A_k}\boldsymbol{S}$, $\boldsymbol{R}\boldsymbol{A_k}\boldsymbol{S}^2$, $\boldsymbol{R}^2\boldsymbol{A_k}\boldsymbol{S}^*$, $\boldsymbol{R}^2\boldsymbol{A_k}\boldsymbol{S}^{2*}$, $\boldsymbol{R}^2\boldsymbol{S}\boldsymbol{A_k}\boldsymbol{S}^{2*}$	–	Added invariants when including $\nabla \boldsymbol{k}$
FS3	$\frac{1}{2}(\|\boldsymbol{R}\|^2 - \|\boldsymbol{S}\|^2)$	$\|\boldsymbol{S}\|^2$	Ratio of excess rotation rate to strain rate
	k †	$\frac{1}{2}\tilde{u}_i\tilde{u}_i$	Turbulence intensity
	$\min(\frac{\sqrt{k}d}{50\nu}, 2)$	–	Wall-distance based Reynolds number
	$\tilde{u}_k \frac{\partial p}{\partial x_k}$ †	$\sqrt{\frac{\partial p}{\partial x_j}\frac{\partial p}{\partial x_j}\tilde{u}_i\tilde{u}_i}$	Pressure gradient along streamline
	$\frac{k}{\epsilon}$	$\frac{1}{\|\boldsymbol{S}\|}$	Ratio of turbulent time scale to mean strain time scale
	$\sqrt{\frac{\partial p}{\partial x_i}\frac{\partial p}{\partial x_i}}$	$\frac{1}{2}\rho\frac{\partial}{\partial x_k}\tilde{u}_k^2$	Ratio of pressure normal stresses to shear stresses
	$\tilde{u}_i\frac{\partial k}{\partial x_i}$ †	$\lvert\overline{u_j'u_k'}S_{jk}\rvert$	Ratio of convection to production of TKE
	$\|\overline{u_i'u_j'}\|$	k	Ratio of total to normal Reynolds stresses
	$\left\lvert\tilde{u}_i\tilde{u}_j\frac{\partial\tilde{u}_i}{\partial x_j}\right\rvert$ †	$\sqrt{\tilde{u}_l\tilde{u}_l\tilde{u}_i\frac{\partial\tilde{u}_i}{\partial x_j}\tilde{u}_k\frac{\partial\tilde{u}_k}{\partial x_j}}$	Non-orthogonality between velocity and its gradient

ples are taken from the available data with replacement until the original size of the data-set is reached, a number of samples will not be present in the training data-set for each decision tree, called out-of-bag (OoB) samples. These OoB samples can then be used to give a validation error, or OoB error. During training of the trees, this OoB error can be observed to determine when training can be stopped. Using the OoB error is similar to performing N-fold cross validation [16]. Using the OoB error allows us to optimize the number of trees in the forest during training. While hyperparameters of the TBRF were tuned (see Section 3), the algorithm is robust to the choice of hyperparameters. It will work out-of-the-box without much tuning quite well, see the appendix for more details. Compared to neural networks, the random forest algorithm is furthermore easy to train, since one does not have to worry about selecting the appropriate optimization algorithm which has its own set of hyperparameters, and its convergence.

The TBRF algorithm presented here was implemented in python, for the source code see [20].

2.6. Choice of input features

Under the modelling assumption that the Reynolds stress tensor can be well approximated using only the mean stress and rotation tensors, $\boldsymbol{S}$ and $\boldsymbol{R}$, [39], the 5 invariants of the tensor basis (8), namely $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_5)$ are sufficient to describe every possible tensor function. In the context of machine-learning, this choice of input features was made in e.g. [29]. However, if we relax this assumption, then it is reasonable to also use other quantities available in the mean-flow as inputs, provided they are appropriately normalized and Galilean invariant. In particular, in the case of the square duct (see Section 3.1) it was observed that due to the symmetry of the case there are only two distinct "basis functions" defined by $\boldsymbol{\theta}$, and these are not sufficient to accurately describe the DNS anisotropy tensor for this case.

Therefore here we will use the full set of invariants derived from $\boldsymbol{S}$, $\boldsymbol{R}$, and $\nabla\boldsymbol{k}$ from Wang et al. [51]. In order to use the turbulent kinetic energy gradient it is first normalized using $\sqrt{k}/\epsilon$, and then transformed to an antisymmetric tensor:

$$\boldsymbol{A}_k = -\boldsymbol{I} \times \nabla k. \tag{13}$$

Furthermore nine extra scalar features which are more physically interpretable, such as the wall-distance based Reynolds number are used, which were obtained from Wu et al. [55] (which were in turn based on those presented in [30]). All features which are used are presented in Table 1, where feature set 1 (FS1) is based on $\boldsymbol{S}$ and $\boldsymbol{R}$ only, feature set 2 (FS2) additionally $\boldsymbol{A}_k$, and feature set 3 (FS3) are additionally the features from Wu et al. [55]. For the features in FS3 an normalization factor is included, whereas the tensors in FS1 and FS2 are normalized using k and ϵ. Note that all features in FS3 are rotationally invariant, but some (or their normalization factors) are not Galilean invariant as they include terms depending on the velocity of the flow – the distinction is marked with a † in the table.

2.7. Propagation of the predicted anisotropy tensor

The open source CFD toolbox OpenFOAM was used to calculate RANS flow fields in this work. The $k-\omega$ turbulence closure model was used, together with the second-order accurate SIMPLE (Semi-Implicit Method for Pressure Linked Equation) scheme.

Simply setting the prediction of the anisotropy tensor $\boldsymbol{b}_{\mathrm{ML}}$ in the momentum equation adversely affects the numerical stability of the solver. As already shown in [56], treating the Reynolds stress as an explicit source term in the RANS equations can lead to an ill-conditioned model. Two main strategies are used here to improve stability: (a) under-relaxing $\boldsymbol{b}_{\mathrm{ML}}$ against the Boussinesq $\boldsymbol{b}_{\mathrm{B}} := \nu_t\hat{\boldsymbol{S}}$ with a relaxation parameter $\gamma \in [0, 1]$, and (b) simultaneously solving a modified k-equation to obtain a turbulence kinetic energy corresponding to the modified anisotropy.

In detail, the incompressible Reynolds-averaged Navier-Stokes equations are

$$\frac{\partial\bar{\boldsymbol{u}}}{\partial t} + \bar{\boldsymbol{u}}\cdot\nabla\bar{\boldsymbol{u}} = \nabla\cdot\left[-\bar{p} + \nu\hat{\boldsymbol{S}} - \boldsymbol{\tau}\right] \tag{14}$$

where ν is the molecular viscosity. The prediction $\boldsymbol{b}_{\mathrm{ML}}$ is introduced into the momentum equation, by modelling $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} \simeq \boldsymbol{\tau}_{\mathrm{ML}}(\gamma) := \frac{2}{3}k\boldsymbol{I} + 2k[(1-\gamma)\boldsymbol{b}_{\mathrm{B}} + \gamma\boldsymbol{b}_{\mathrm{ML}}]. \tag{15}$$

The blending parameter γ starts at 0 and is gradually increased during the simulation, i.e. a continuation method, see e.g. [22]. A linear ramp based on the iteration count n is used:

$$\gamma_n = \gamma_{\max}\min\left\{1, \frac{n}{n_{\max}}\right\},$$

where $\gamma_{\max} \geq 0.8$ is achieved in all test-cases presented here, and $n_{\max}$ is the iteration count, after which γ is fixed. Sufficient itera-

tions are performed after this point to achieve solver convergence. A lower value for γ means that the linear eddy viscosity assumption becomes more dominant, resulting in a more stable solution, but impairing the accuracy of the solved mean velocity as already noted in [56]. Here, $\gamma_{\max}$ was incremented in steps of 0.1 until the solver became unstable, yielding a value of $\gamma_{\max} = 0.8$. As this choice is ad hoc, further work related to this topic is necessary.

Furthermore, the turbulent kinetic energy in (15) is obtained by solving a version of the $k-\omega$ k-equation, in which the production term is modified to be consistent with the predicted Reynolds stress in the momentum equation. The standard production term

$$\mathcal{P} = -\boldsymbol{\tau} : \nabla \bar{\boldsymbol{u}}, \tag{16}$$

is approximated in the $k-\omega$ model by replacing $\boldsymbol{\tau}$ with its Boussinesq approximation [54]. Here we use the model $\boldsymbol{\tau}_{\mathrm{ML}}$ from (15) instead, including the blending with γ.

With these modifications, the solver converges for $\boldsymbol{b}$-tensors originating from DNS, TBNN and TBRF.

3. Results

We compare predictions of the TBRF algorithm just described, with baseline RANS ($k-\omega$), DNS/LES references (withheld reference data), and the TBNN algorithm with the same feature sets as TBRF. Data for training and predicting comes from five flow cases which will be discussed briefly in Section 3.1. Predictions of the anisotropy tensor itself will be presented in Section 3.2; corresponding mean-flow predictions are presented in Section 3.3.

3.1. Flow cases

Five flow cases are used in the framework to train and test the machine learning algorithms. For all flow cases DNS data or highly resolved LES data was available. The mean flows for the reference DNS/LES solutions are presented in Fig. 4. The flow cases are:

(a) **Periodic hills (PH)**: Five Reynolds numbers are available in the DNS/LES data-set from Breuer et al. [5], ranging from $Re = 700$ to $Re = 10595$ based on the bulk velocity at the inlet and the hill height.
(b) **Converging-diverging channel (CD)**: The DNS data for comes from Laval and Marquillie [24] at $Re = 12600$ based on the channel half-height and the maximum velocity at the inlet.
(c) **Curved backward-facing step (CBFS)**: The Reynolds number available is $Re = 13700$ based on the step height and the center-channel inlet velocity, with highly resolved LES data from Bentaleb et al. [3].
(d) **Backward-facing step (BFS)**: $Re = 5100$ based on the step height and free stream velocity. The corresponding DNS simulation can be found in [26].
(e) **Square duct (SD)**: Data-sets at multiple Reynolds numbers are available from Pinelli et al. [38], with a total of sixteen ranging from $Re = 1100$ to $Re = 3500$ based on the duct semi-height and the bulk velocity.

The first four aforementioned cases feature flow separation and subsequent reattachment. Recirculation bubbles, non-parallel shear layers, and mean-flow curvature are all known to pose challenges for RANS based turbulence models. The square duct flow case is symmetric; Fig. 4(e) only presents the upper right quadrant of the duct, where the flow in the duct moves out-of-plane. Prandtl's secondary motion of the second kind is visible, driven by turbulence anistropy. As such it is not captured at all by linear eddy-viscosity models, which makes them ideal for isolating effects of nonlinear modelling [37]. For all cases mesh independence studies were performed for the RANS simulations, and meshes were chosen such that discretization error was a small fraction of the turbulence modelling error.

3.2. Anisotropy tensor predictions

In this section we examine the quality with which the Reynolds anisotropy tensor is reproduced by the TBRF. We compare by examining (a) individual tensor components, and (b) eigenvalues in the barycentric map.

Four test-cases are presented in Table 2, including details of training and prediction flows. In this table the names of the flow cases have been abbreviated, and the number behind the abbreviation indicates the Reynolds number. The table also presents the number of samples used for training, N_{sample} (randomly sampled from the total data-set), and the number of usable features present in the training sets, N_{feature}. From the available features the ones with low variance ($< 1 \times 10^{-4}$) were discarded, as these either did not contain any information at all, or were largely spurious. The starting feature sets (FS) used are those specified in Table 1. For all cases the $k-\omega$ turbulence model was used for the RANS simulations. Hyperparameters of the TBRF were tuned for cases C1, C2, and C4 using a validation set consisting of PH2800 and SD3200. A total of 100 TBDT's were used to make the predictions. From the 17 available features 11 were randomly selected for calculating each optimal split in the TBDT. The leaf nodes were set to have a minimum of 9 samples, and the regularization factor Γ was set to 1×10^{-12}. For case C3 the same setting were used, except that the trees were fully grown (i.e. each leaf node consists of one sample), and all features were used to calculate the optimal split.

First prediction for the curved backward facing step will be presented (C1), which is relatively similar to the training cases for which reliable data was available (periodic hills and the converging-diverging channel). Next, results for the backward facing step will be presented (C2), which features stronger separation than to the training cases and will therefore feature more extrapolation. Lastly, results for the square duct will be presented. A comparison will be made for the case using only features based on $\boldsymbol{S}$ and $\boldsymbol{R}$ as was also done in [29] (C3), and a case where all available features are used (C4).

3.2.1. Curved backward facing step

For the curved backward facing step (case C1 in Table 2), Fig. 5 presents the four non-zero unique components of $\boldsymbol{b}$, as given by the LES data, the ($k-\omega$), and the TBRF and TBNN algorithms. Taking LES as a reference, RANS only gives acceptable predictions for the $[\boldsymbol{b}]_{12}$ component. By the Boussinesq assumption, results will only be acceptable where the turbulence anisotropy tensor is aligned with the mean rate of strain tensor, which is empirically not a good assumption in the majority of the domain. In contrast, both machine learning algorithms give reasonable predictions of the tensor field in all components, and predictions are relatively smooth. In particular, features on top of the step the $[\boldsymbol{b}]_{11}$ component is captured qualitatively by the TBRF and TBNN algorithms, as well as the $[\boldsymbol{b}]_{33}$ component after the step.

More revealing is a plot of the stress types in the domain using the RGB map, Fig. 6. The LES data shows 1-component turbulence on top of the step, which is transported into the flow, beyond the location at which the shear layer separates. In [3] it is noted that production of the streamwise fluctuation is strongly increased at the shear-layer separation location, leading to additional 1-component turbulence. As this shear layer gains distance from the wall, the turbulence rapidly evolves towards the 3-component state due to the redistribution process. On the curved part of the step and the bottom wall, 2-component turbulence can be observed. In the remainder of the domain, 3-component turbulence

Table 2
Data-sets used for training and testing. PH = Periodic Hills; CD = Converging-Diverging channel; CBFS = Curved Backward Facing Step; BFS = Backward Facing Step; and SD = Square Duct.

Case nr.	Training	Prediction	N_{sample}	N_{feature}	Feature sets
C1	PH5600, PH10595, CD12600	CBFS13700	21,000	17	FS1, FS2, FS3
C2	PH5600, PH10595, CD12600	BFS5100	21,000	17	FS1, FS2, FS3
C3	PH5600, PH10595, CD12600	SD3500	21,000	5	FS1
C4	PH5600, PH10595, CD12600	SD3500	21,000	17	FS1, FS2, FS3

dominates, in the interior of the channel, and in the center of the recirculation region.

The RANS simulation is only capable of predicting plane-strain (c.f. Fig. 3), and predicts turbulence mainly in the 3-component region. The effect of the walls on the turbulence anisotropy is completely missed. In contrast, the TBRF algorithm accurately captures the turbulent state as given by the LES data: 1-component turbulence can be seen on top of the hill and at the separation location, it accurately predicts the 2-component state on the curved part of the walls and on the bottom wall after the step, and 3-component turbulence can be observed in the recirculation region. Some noise is visible however, most notably around $x/h = 0.0$ to $x/h = 1.0$ away from the wall. The TBNN algorithm captures the different types of turbulence accurately as well. Close to the wall on top of the step it captures the 1-component turbulence a bit less well than the TBRF algorithm, and some spurious patterns are visible above the step and close to the lower wall around $x/h = 6.0$.

To better quantify the accuracy of reconstruction, three sections through the flow domain are plotted in the barycentric map. These sections are located at $x/h = 2$, $x/h = 3$, and $x/h = 4$, which which are at the front, middle, and aft part of the separated region. The first section at $x/h = 2$ ranges from $y/h = 0.5$ to $y/h = 1.5$, the other two sections range from $y/h = 0.0$ to $y/h = 1.5$. Results are presented in Fig. 7. As can be seen both machine learning algorithms reproduce quite closely the LES reference data. They accurately capture the 2-component turbulence close to the wall, and move towards the 3-component corner when moving away from the wall. Discrepancies can be seen when moving upwards past to the wake to the channel center, where the LES data indicates a move towards axisymmetric expansion, which is less strongly represented by the ML algorithms.

3.2.2. Backward facing step

We consider case C2 (c.f. Table 2). From Le and Moin [25] DNS data is available for five different sections at specified x/h locations for the Reynolds stresses and velocities (h is the step-height). Locations on the barycentric map for these five sections are plotted in Fig. 8. The sections range from $y/h = -1$ (bottom wall) to $y/h = 0$ (the location of the step).

Results are similar to those of the CBFS: the machine-learning algorithms are able to give a qualitatively accurate prediction of the turbulence character, with some quantitative discrepancies. For $x/h = 4$ and $x/h = 6$ predictions close to the wall are more accurate for TBRF than TBNN. The situation is reversed for $x/h = 10, 15, 19$, where TBNN slightly outperforms, at the cost of some unrealizable predictions closest to the wall. In all our studies, we have never observed unrealizable predictions from TBRF, despite no explicit realizability constraint being imposed on the method.

Moving away from the wall into the shear layer TBRF erroneously heads too far back towards the two-component boundary at the sections closest to the step. The reason for this is unclear, at similar (shear-layer) locations in the training flows, the turbulence does not exhibit such behaviour. Furthermore TBNN is reasonably accurate here. Diagnostic tools are needed, and will be a focus of future research. Nonetheless at the section further from the step, both ML methods perform well.

Table 3
RMSE of TBRF and TBNN $[\boldsymbol{b}]_{ij}$ predictions, for the square duct flow case.

Case	TBRF	TBNN
C3	0.0995	0.0871
C4	0.0521	0.0681

Table 4
Backward facing step, reattachment point locations.

Model	x_{reattach} [x/h]
RANS	5.45
RANS+$b_{ij,\text{TBRF}}$	6.32
DNS [26]	6.28
Experiment [18]	6.0 ± 0.15

3.2.3. Square duct

The local stress type for the square duct was already shown in Fig. 3; individual components of the anisotropy tensor shown in Fig. 9. In both cases anistropy is visualized for DNS, RANS ($k-\omega$), TBRF and TBNN predictions. In Fig. 3 ML results are only shown for case C4 (17 features); in Fig. 3 additionally case C3 is shown. Note that these are challenging cases due to the substantial differences between the training and prediction flows.

As expected, the Boussinesq model yields non-zero predictions only for $[\boldsymbol{b}]_{12}$ and $[\boldsymbol{b}]_{13}$ – though these are relatively well predicted. Anistropy is confined to the 3-component corner, on the plane-strain line. Examining the predictions of ML, it can generally seen that the introduction of extra features has significantly more effect than the choice of neural-networks versus random-forests. For example, looking at $[\boldsymbol{b}]_{11}$, the anisotropy of the Reynolds stress is not captured close to the walls for C3, whereas it is present in C4. The magnitude of $[\boldsymbol{b}]_{12}$ and $[\boldsymbol{b}]_{13}$ is underpredicted, independently of the ML method, but improved in case C4 compared to C3. Similarly in all cases the magnitude of $[\boldsymbol{b}]_{23}$ is over-predicted by ML, but the magnitude of the over-prediction is less in case C4. To quantify these observations, the root mean square error (RMSE) of the anisotropy tensor with respect to the DNS data is given in Table 3. The RMSE's are lower when introducing more features, for both algorithms. This can largely be explained by visualizing the shapes of the input features in case C3. Doing this it can be observed that, of the 5 features, 3 are approximately scaled versions of the other 2 – effectively reducing the input space to two-dimensions. This explains the difficulty nonlinear eddy-viscosity models based on only $\boldsymbol{S}$ and $\boldsymbol{R}$, have in reproducing the magnitude of the secondary flow in the square duct.

3.3. Anisotropy tensor propagation

This section presents flow fields obtained by propagating the predicted anisotropy tensors for the square duct flow and the backward facing step. Predictions for the anisotropy tensor were propagated using the stabilized solver presented in Section 2.7.

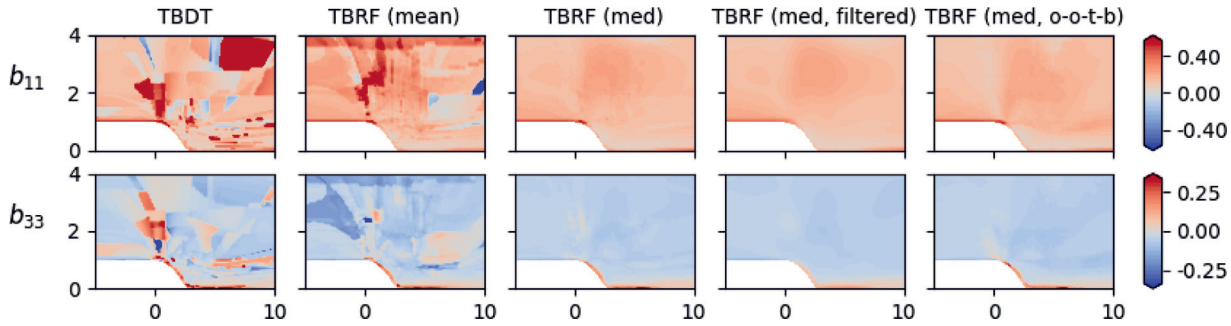


Fig. 13. Illustration of the robustness of the TBRF algorithm.

3.3.1. Square Duct

Two sections in the square duct will be analyzed with respect to the in-plane mean velocity magnitude (i.e. indicating the magnitude of the secondary flow). Fig. 10 presents the velocity magnitude for sections located at $y/h = 0.5$ and $y/h = 0.8$. DNS from Pinelli et al. [38] is used as a reference. In order to verify the propagation method in isolation (without predicting anistropy), the anisotropy tensor obtained directly from DNS ($b_{ij,\mathrm{DNS}}$) is propagated, see the gray lines. This is a "best case scenario" where the anisotropy tensor is assumed to be perfect (up to statistical convergence of the DNS). The mean-flow field as obtained by propagating the predictions from the TBRF algorithm ($b_{ij,\mathrm{TBRF}}$, see column 5 of Fig. 9) is indicated by the red lines. Furthermore, results from the quadratic eddy viscosity model of Shih et al. [44], and the cubic eddy viscosity model of Lien et al. [27] are presented. Since the linear eddy-viscosity model does not yield any secondary flow at all, this result is omitted.

When examining the results of Fig. 10, the story is broadly the same for $y/h = 0.5$ and $y/h = 0.8$. Mean-velocity fields obtained using $b_{ij,\mathrm{DNS}}$ broadly reproduce the DNS mean velocity, both in amplitude and location of key features, with the best fit near the wall ($z = 1$), and the worst near the channel centerline ($z = 0$). Subsequently approximating $b_{ij,\mathrm{DNS}}$ by $b_{ij,\mathrm{TBRF}}$ causes additional errors, but theses errors are of similar magnitude to the errors already made in the propagation. In particular, key features are correct, and amplitudes are appropriate. What is also clear however, is that predictions are still far more accurate than both non-linear eddy-viscosity models. The model from Shih et al. [44] is able to predict the location of the peaks of the in-plane flow magnitude quite accurately, but significantly underpredicts the overall magnitude. Predictions by the cubic eddy-viscosity model from Lien et al. [27] are far off overall.

3.3.2. Backward facing step

Fig. 11 presents streamlines in the flow field for the backward facing step, with results from $k-\omega$ RANS, the propagated velocity field using the predicted anisotropy tensor from the TBRF algorithm ($b_{ij,\mathrm{TBRF}}$), and DNS data from Le et al. [26]. The size of the recirculation region is more correctly predicted for the propagated velocity field using $b_{ij,\mathrm{TBRF}}$ compared to the baseline RANS simulation. Further away from the wall the solver using $b_{ij,\mathrm{TBRF}}$ does not introduce spurious effects and results are similar to the baseline RANS simulation. The reattachment point locations (x_{reattach}) for all three cases are presented in Table 4. A significant improvement is shown for the propagated velocity field compared to the baseline RANS simulation.

The skin friction coefficients from the RANS simulation and the propagated flow field using $b_{ij,\mathrm{TBRF}}$ are compared to experimental data from Jovic and Driver [18] in Fig. 12. The propagated flow field shows a very close match to the experimental data, and the majority of results fall within the error bounds given by the experiment ($\pm\ 0.0005 c_f$). The reattachment point of the propagated flow field (6.32) compares favourably to the experimentally found reattachment point (6.0 ± 0.15).

4. Conclusions

In this work, a novel random forest algorithm was introduced for RANS turbulence modeling, to model the Reynolds stress anisotropy tensor. The algorithm was trained using invariant input features from several RANS ($k-\omega$) flow fields, and the corresponding responses for the anisotropy tensor from DNS or highly-resolved LES data. Galilean invariance of the predicted anisotropy tensor is ensured by making use of a tensor basis, derived in [39]. The new random forest algorithm is called the Tensor-Basis Random Forest (TBRF) algorithm, similarly to the Tensor-Basis Neural Network from Ling et al. [29] from which it was inspired. Robust predictions of the Reynolds-Stress anisotropy tensor are obtained by taking the median of the Tensor-Basis Decision Tree (TBDT) predictions inside the TBRF.

Predictions for the Reynolds-stress anisotropy tensor were presented for the square duct flow case, curved backward-facing step, and backward-facing step. Improvement is observed with respect to the baseline $k-\omega$ simulations, and the TBRF algorithm performs on par with the TBNN algorithm. Compared to TBNN, the TBRF algorithm is relatively easy to implement and train (one does not have to think about matters such as the optimization algorithm used to tune the neural network weights and its convergence); the out-of-bag samples from the decision trees allow for a natural way to quantify the validation error during training and thus selecting the amount of trees to be used in the random forest. The few remaining hyperparameters are quite robust: the TBRF works well out-of-the-box even when using standard hyperparameter settings (fully grown trees, using all available features for creating the decision tree splits).

A custom solver for propagating the anisotropy tensor was introduced, which blends the predictions for the anisotropy tensor with a $k-\omega$ turbulence model. This solver greatly increases numerical stability of the propagation. Propagations for the square duct flow case and backward facing step are presented, which show a close match with respect to corresponding DNS and experimental data-sets.

A possibility for future work might be using the TBRF for quantifying uncertainty of the predictions as well. For every location in the flow domain the trees in the random forest can be analyzed for their variance, which would make it possible to use an anisotropy eigenvalue perturbation methodology to quantify uncertainty such as proposed in [13]. In order to achieve meaningful bounds for uncertainty of the predictions, one could look at e.g. Bayesian Additive Regression Trees [8], or jackknife/infinitesimal jackknife methods [15], or modify the random forest algorithm itself for meaningful uncertainty bounds, see e.g. [33,34]. It was observed, that often

the individual decision trees show a high variance of the prediction, when the prediction itself is relatively poor [19].

Future work should further investigate the performance of the TBRF algorithm and the relaxation solver on more complex flow cases. Only a number of idealized canonical flow cases were considered here, it would be interesting to see how well the algorithm does in for example highly detached flows, how well the algorithm is able to extrapolate to higher Reynolds numbers, and whether it could be used for unsteady RANS flows as well.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

CRediT authorship contribution statement

Mikael L.A. Kaandorp: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Richard P. Dwight:** Conceptualization, Methodology, Supervision, Writing - original draft.

Acknowledgments

The authors would like to thank Javier Fatou Gómez for supplying the OpenFOAM continuation solver used to propagate the Reynolds stresses into the flow field; also Julia Ling for providing her implementation of TBNN and support for this work.

Appendix. TBRF implementation details

Decision trees base their predictions on a series of if-then tests on the input. Random forests consist of collections of decision trees with some randomized component differentiating trees. Multiple decision tree algorithms exist, of which the CART (Classification And Regression Tree) algorithm is used as a starting point here.

As mentioned in Section 2.5, the feature space is recursively split into two bins, R_R and R_L. The splitting is performed greedily, with each split selected to minimize the mismatch between the response $\boldsymbol{y}$, and the best constant approximation of the response in both bins. Specifically for each split we solve [16]:

$$\min_{j,s}\left[\min_{c_L\in\mathbb{R}}\sum_{x_i\in R_L(j,s)}(\boldsymbol{y}_i-c_L)^2+\min_{c_R\in\mathbb{R}}\sum_{x_i\in R_R(j,s)}(\boldsymbol{y}_i-c_R)^2\right], \tag{17}$$

where $\boldsymbol{y}_i$ denotes the response at $\boldsymbol{X}_i$. Finding constants $c_L, c_R \in \mathbb{R}$ amounts to averaging $\boldsymbol{y}$ within R_L and R_R respectively, effectively minimizing the variance in both bins. Starting from the full dataset the same method is then applied to R_L and R_R in a recursive fashion. The procedure is terminated either at a specified maximum branching depth, or a minimum number of samples per bin.

The new TBDT algorithm is comparable with the CART decision tree algorithm, but instead of approximating the response with constant values c_L and c_R, the algorithm finds a constant value for each of the tensor basis coefficients $g^{(m)}$ in (8), chosen to minimize the mismatch between this expression and the anisotropy tensor from DNS. Specifically we solve

$$\min_{j,s}\left[\min_{g_L^{(m)}\in\mathbb{R}^{10}}\sum_{x_i\in R_L(j,s)}\left\|\sum_{m=1}^{10}\boldsymbol{T}_i^{(m)}g_L^{(m)}-\boldsymbol{b}_i\right\|_F^2\right.$$
$$\left.+\min_{g_R^{(m)}\in\mathbb{R}^{10}}\sum_{x_i\in R_R(j,s)}\left\|\sum_{m=1}^{10}\boldsymbol{T}_i^{(m)}g_R^{(m)}-\boldsymbol{b}_i\right\|_F^2\right], \tag{18}$$

where i indexes the samples, $\boldsymbol{b}_i$ is the DNS/LES anisotropy tensor, and $g_L^{(m)}$ and $g_R^{(m)}$ are the tensor basis coefficients in the left and right bins respectively. The norm used is the Frobenius norm:

$$\|\boldsymbol{A}\|_F := \sqrt{\sum_{i,j}[\boldsymbol{A}]_{ij}^2} = \sqrt{\mathrm{tr}(\boldsymbol{A}^T\boldsymbol{A})}, \tag{19}$$

where $\boldsymbol{A}$ can be an arbitrary second-order tensor. This norm is chosen for its invariance to unitary transformations – in particular rotations. Now finding $g_L^{(m)}$ and $g_R^{(m)}$ amounts to solving two least-squares problems. This must be repeated for every j and for every optimization iteration of s. As for CART, (18) is repeated for each bin, until a stopping criterion is reached.

Explicitly, by flattening the tensor at each point, and defining:

$$\hat{\boldsymbol{T}}_i = \begin{bmatrix} [\boldsymbol{T}_i^{(1)}]_{11} & [\boldsymbol{T}_i^{(2)}]_{11} & \cdots & [\boldsymbol{T}_i^{(10)}]_{11} \\ [\boldsymbol{T}_i^{(1)}]_{12} & [\boldsymbol{T}_i^{(2)}]_{12} & \cdots & [\boldsymbol{T}_i^{(10)}]_{12} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{T}_i^{(1)}]_{33} & [\boldsymbol{T}_i^{(2)}]_{33} & \cdots & [\boldsymbol{T}_i^{(10)}]_{33} \end{bmatrix}, \quad \hat{\boldsymbol{b}}_i = \begin{bmatrix} [\boldsymbol{b}_i]_{11} \\ [\boldsymbol{b}_i]_{12} \\ \vdots \\ [\boldsymbol{b}_i]_{33} \end{bmatrix}, \tag{20}$$

each of the minimization problems over $\boldsymbol{g}$ becomes $\min_{\boldsymbol{g}} J$ where

$$J = \sum_{i=1}^{N}\|\hat{\boldsymbol{T}}_i\boldsymbol{g}-\hat{\boldsymbol{b}}_i\|^2, \tag{21}$$

with solution

$$\boldsymbol{g} = \left(\sum_{i=1}^{N}\hat{\boldsymbol{T}}_i^T\hat{\boldsymbol{T}}_i\right)^{-1}\left(\sum_{i=1}^{N}\hat{\boldsymbol{T}}_i^T\hat{\boldsymbol{b}}_i\right). \tag{22}$$

which can be solved separately for R_L and R_R to obtain $g_L^{(m)}$ and $g_R^{(m)}$. The overall cost of this algorithm (as for CART) is domainated by sorting the data-values with respect to coordinate j. This cost is $\mathcal{O}(N\log N)$ in the number of data-values, leading to an overall cost of $\mathcal{O}(N\log^2 N)$. Unlike training neural networks, this procedure is fast, robust, easy to implement, and independent of any starting guess.

Due to the redundancy of the tensor-basis for any given sample $i \in \{1,\dots,N\}$, this problem can become ill-posed, especially towards the leaves of the tree, when only a few samples remain in a bin. Therefore some L^2-regularization is added to J with coefficient $\Gamma \in \mathbb{R}^+$, c.f. ridge regression. To summarize, (22) is modified to

$$\boldsymbol{g} = \left(\sum_{i=1}^{N}\hat{\boldsymbol{T}}_i^T\hat{\boldsymbol{T}}_i+\Gamma\boldsymbol{I}\right)^{-1}\left(\sum_{i=1}^{N}\hat{\boldsymbol{T}}_i^T\hat{\boldsymbol{b}}_i\right). \tag{23}$$

In practice it was observed that by taking the median of all decision trees instead of the mean (see Section 2.5), this already provides a lot of robustness, and that Γ can be set to a very low value in general. The value of Γ in the random forest was tuned using validation data-sets, see Section 3.2.

One important difference of the TBRF over the standard random forest is the fact that it effectively does not directly predict the final outcome, but the coefficients $\boldsymbol{g}$, which multiply the basis tensors $\boldsymbol{T}$. Unlike the standard random forest, this means that the values for the final predictions do not have to lie in-between the values of the points used for training. Decision trees are well known to be sensitive to small changes in the data, and this manifested during testing as highly irregular and inconsistent predictions in small regions of the spatial domain. For a sample TBDT prediction

of two components of the anisotropy tensor for the curved backward facing step (further introduced in Section 3.1), where these inconsistencies are clearly visible, see the first column in Fig. 13. Both pruning and regularization were applied to reduce these inconsistencies. While regularization fixed the problem to some extent, it worsened predictions in certain regions of the flow, since this did not allow the coefficients $\boldsymbol{g}$ to vary sufficiently. Instead, it proved to be more successful to take the median of the trees in the random forest instead of the mean (as for example investigated in [43]), since this removed sensitivity to outliers in the predictions. A comparison between taking the mean and median is presented in column 2 and 3 of Fig. 13. Column 4 presents the results after applying the Gaussian smoothing as described in Section 2.5. As described in Section 2.5, the TBRF algorithm will work out-of-the-box without much tuning quite well, as demonstrated in column 5 of Fig. 13. Predictions are shown for a TBRF with standard settings (fully grown decision trees, using all features for creating the splits), and a arbitrarily low regularization factor of $\Gamma = 1 \times 10^{-15}$. As can be seen the anisotropy predictions are quite insensitive to the set of hyperparameters when comparing to the predictions from the tuned TBRF in column 3.

References

[1] Akolekar HD, Sandberg RD, Hutchins N, Michelassi V, Laskowski G. Machine-learnt turbulence closures for lpt's with unsteady inflow conditions. In: Proceedings of the 15th international symposium on unsteady aerodynamics, aeroacoustics & aeroelasticity of turbomachines; 2018.

[2] Banerjee S, Krahl R, Durst F, Zenger C. Presentation of anisotropy properties of turbulence, invariants versus eigenvalue approaches. J Turbul 2007;8(32). doi:10.1080/14685240701506896.

[3] Bentaleb Y, Lardeau S, Leschziner M. Geometric properties of particle trajectories in turbulent flows. J Turbul 2012;13:1–28. doi:10.1080/14685248.2011.637923.

[4] Brent RP. An algorithm with guaranteed convergence for finding a zero of a function. Algorithms for minimization without derivatives. Prentice-Hall; 1973. ISBN 0-13-022335-2.

[5] Breuer M, Peller N, Rapp C, Manhart M. Flow over periodic hills - numerical and experimental study in a wide range of Reynolds numbers. Comput Fluids 2009;38(2):433–57. doi:10.1016/j.compfluid.2008.05.002.

[6] Chen X-W, Lin X. Big data deep learning: challenges and perspectives. IEEE Access 2014;2:514–25. doi:10.1109/ACCESS.2014.2325029.

[7] Cheung SH, Oliver TA, Prudencio EE, Prudhomme S, Moser RD. Bayesian uncertainty analysis with applications to turbulence modeling. Reliab Eng Syst Saf 2011;96(9):1137–49. doi:10.1016/j.ress.2010.09.013.

[8] Chipman HA, George EI, McCulloch RE. Bart: Bayesian additive regression trees. Ann Appl Stat 2010;4(1):266–98. doi:10.1214/09-AOAS285.

[9] Craft TJ, Launder BE, Suga K. Development and application of a cubic eddy-viscosity model of turbulence. Int J Heat Fluid Flow 1996;17(2):108–15. doi:10.1016/0142-727X(95)00079-6.

[10] Duraisamy K, Iaccarino G, Xiao H. Turbulence modeling in the age of data. Annu Rev Fluid Mech 2019;51:1–23. doi:10.1146/annurev-fluid.

[11] Duraisamy K, Zhang ZJ, Anand PS. New approaches in turbulence and transition modeling using data-driven techniques. In: 53rd AIAA Aerospace sciences meeting; 2015. p. 1–14. doi:10.2514/6.2015-1284.

[12] Edeling WN, Cinnella P, Dwight RP. Predictive RANS simulations via Bayesian model-scenario averaging. J Comput Phys 2014;275(October):65–91. doi:10.1016/j.jcp.2014.06.052.

[13] Emory M, Larsson J, Iaccarino G. Modeling of structural uncertainties in Reynolds-averaged Navier-Stokes closures. Phys Fluids 2013;25(11). doi:10.1063/1.4824659.

[14] Esposito F, Malerba D, Semeraro G. A comparative analysis of methods for pruning decision trees. IEEE Trans Pattern Anal Mach Intell 1997;19(5):476–91. doi:10.1109/34.589207.

[15] Giordano R, Stephenson W, Liu R, Jordan MI, Broderick T. A Swiss army infinitesimal jackknife. In: Proceedings of the 22nd international conference on artificial intelligence and statistics (AISTATS), Naha, Okinawa, Japan; 2019.

[16] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. second ed. Springer; 2008.

[17] Higham NJ. Functions of matrices: theory and computation. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2008.

[18] Jovic S, Driver D. Backward-facing step measurements at low reynolds number. NASA Technical Memorandum 108807; 1994.

[19] Kaandorp M. Machine learning for data-driven RANS turbulence modelling; 2018. Master Thesis. http://resolver.tudelft.nl/uuid:f833e151-7c0f-414c-8217-5af783c88474.

[20] Kaandorp M, Dwight RP. Tensor basis random forest github repository; 2019. doi:10.6084/m9.figshare.11352137.v1.

[21] Kingma DP, Ba JL. Adam: a method for stochastic optimization. In: Proceedings of the 3rd international conference on learning representations (ICLR); 2015. https://arxiv.org/pdf/1412.6980.pdf.

[22] Knoll DA, Keyes DE. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. J Comput Phys 2004;193:357–97. doi:10.1016/j.jcp.2003.08.010.

[23] Launder B, Spalding D. The numerical computation of turbulent flows. Comput Methods Appl Mech Eng 1974;3(2):269–89. doi:10.1016/0045-7825(74)90029-2.

[24] Laval J-P, Marquillie M. Direct numerical simulations of converging-diverging channel flow. In: Progress in wall turbulence: understanding and modeling; 2011. p. 203–9. ISBN 9789048196029. doi:10.1007/978-90-481-9603-6.

[25] Le H, Moin P. Direct numerical simulation of turbulent flow over a backward–facing step. Stanford University, Center for Turbulence Research, Annual Research Briefs; 1992. p. 161–73.

[26] Le H, Moin P, Kim J. Direct numerical simulation of turbulent flow over a backward-facing step. J Fluid Mech 1997;330:349–74.

[27] Lien F, Chen W, Leschziner MA. Low-Reynolds-number eddy-viscosity modelling based on non-linear stress-strain/vorticity relations. In: Engineering turbulence modelling and experiments 3. Elsevier Science; 1996. p. 91–100.

[28] Ling J, Jones R, Templeton J. Machine learning strategies for systems with invariance properties. J Comput Phys 2016;318:22–35. doi:10.1016/j.jcp.2016.05.003.

[29] Ling J, Kurzawski A, Templeton J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J Fluid Mech 2016;807:155–66. doi:10.1017/jfm.2016.615.

[30] Ling J, Templeton J. Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. Phys Fluids 2015;27(8). doi:10.1063/1.4927765.

[31] Lumley JL. Computational modeling of turbulent flows. Adv Appl Mech 1979;18:123–76. doi:10.1016/S0065-2156(08)70266-7.

[32] Lumley JL, Newman GR. The return to isotropy of homogeneous turbulence. J Fluid Mech 1977;82(1):161–78. doi:10.1017/S0022112077000585.

[33] Meinshausen N. Quantile regression forests. J Mach Learn Res 2006;7:983–99.

[34] Mentch L, Hooker G. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. J Mach Learn Res 2016;17:1–41.

[35] Murphy KP. Machine learning: a probabilistic perspective. The MIT Press; 2012.

[36] Parish EJ, Duraisamy K. A paradigm for data-driven predictive modeling using field inversion and machine learning. J Comput Phys 2016;305:758–74. doi:10.1016/j.jcp.2015.11.012.

[37] Pecnik R, Iaccarino G. Predictions of turbulent secondary flows using the v2 - f model. Stanford University, Center for Turbulence Research, Annual Research Briefs; 2007. p. 333–43.

[38] Pinelli A, Uhlmann M, Sekimoto A, Kawahara G. Reynolds number dependence of mean flow structure in square duct turbulence. J Fluid Mech 2010;644:107. doi:10.1017/S0022112009992242.

[39] Pope SB. A more general effective-viscosity hypothesis. J Fluid Mech 1975;72(2):331–40. doi:10.1017/S0022112075003382.

[40] Pope SB. Turbulent flows. Cambridge University Press; 2000.

[41] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. MIT Press; 2006.

[42] Reynolds O. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. Philos Trans R Soc Lond A 1895;186:123–64.

[43] Roy M-H, Larocque D. A methodology to evaluate statistical errors in DNS data of plane channel flows. J Nonparametr Stat 2012;24:993–1006. doi:10.1080/10485252.2012.715161.

[44] Shih T-h, Zhu J, Lumley JL. A realizable Reynolds stress algebraic equation model. NASA Technical Memorandum 105993; 1993.

[45] Singh AP, Medida S, Duraisamy K. Machine learning-augmented predictive modeling of turbulent separated flows over airfoils; 2016. doi:10.1016/j.jcp.2015.11.012.

[46] Slotnick J, Khodadoust A, Alonso J, Darmofal D, Gropp W, Lurie E, et al. CFD vision 2030 study: a path to revolutionary computational aerosciences NASA Technical Report NASA/CR-2014-21878; 2014. doi:10.1017/CBO9781107415324.004.

[47] Speziale CG, Sarkar S, Gatski TB. Modelling the pressure-strain correlation of turbulence: an invariant dynamical systems approach. J Fluid Mech 1991;227(December):245–72. doi:10.1017/S0022112091000101.

[48] Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58. doi:10.1109/ACCESS.2014.2325029.

[49] Thompson RL, Sampaio LEB, de Bragana Alves FA, Thais L, Mompean G. A methodology to evaluate statistical errors in DNS data of plane channel flows. Comput Fluids 2016;130:1–7. doi:10.1016/j.compfluid.2016.01.014.

[50] Tracey B, Duraisamy K, Alonso JJ. Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. 51st AIAA ASM including the new horizons forum and aerospace exposition; 2013. doi:10.2514/6.2013-259.

[51] Wang J.-X., Wu J., Ling J., Iaccarino G., Xiao H.. A comprehensive physics-informed machine learning framework for predictive turbulence modeling; 2017. http://arxiv.org/abs/1701.07102.

[52] Wang Q, Dow EA. Quantification of structural uncertainties in the k - omega turbulence model. Center of Turbulence Research, Proceedings of the Summer Program; 2010.

[53] Weatheritt J, Sandberg R. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress-strain relationship. J Comput Phys 2016;325:22–37. doi:10.1016/j.jcp.2016.08.015.
[54] Wilcox DC. Formulation of the k-w turbulence model revisited. AIAA J 2008;46(11):2823–38. doi:10.2514/1.36541.
[55] Wu J-L, Wang J-X, Xiao H, Ling J. Physics-informed machine learning for predictive turbulence modeling: a priori assessment of prediction confidence; 2016. http://arxiv.org/abs/1606.07987.
[56] Wu J-L, Xiao H, Sun R, Wang Q. RANS equations with explicit data-driven Reynolds stress closure can be ill-conditioned; 2019. https://arxiv.org/abs/1803.05581.
[57] Xiao H, Wang J-x, Ghanem RG. A random matrix approach for quantifying model-Form uncertainties in turbulence modeling. Comput Methods Appl Mech Eng 2016;313:941–65. doi:10.1016/j.cma.2016.10.025.