Journal of Computational Physics 383 (2019) 125–147

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks

Nicholas Geneva, Nicholas Zabaras *

Center for Informatics and Computational Science, University of Notre Dame, 311 I Cushing Hall, Notre Dame, IN 46556, USA

ARTICLE INFO

Article history:
Received 6 July 2018
Received in revised form 22 October 2018
Accepted 28 January 2019
Available online 1 February 2019

Keywords:
Turbulence
Reynolds-Averaged Navier–Stokes Equations (RANS)
Model form uncertainty
Uncertainty quantification
Bayesian
Deep neural networks

ABSTRACT

Data-driven methods for improving turbulence modeling in Reynolds-Averaged Navier–Stokes (RANS) simulations have gained significant interest in the computational fluid dynamics community. Modern machine learning algorithms have opened up a new area of black-box turbulence models allowing for the tuning of RANS simulations to increase their predictive accuracy. While several data-driven turbulence models have been reported, the quantification of the uncertainties introduced has mostly been neglected. Uncertainty quantification for such data-driven models is essential since their predictive capability rapidly declines as they are tested for flow physics that deviate from that in the training data. In this work, we propose a novel data-driven framework that not only improves RANS predictions but also provides probabilistic bounds for fluid quantities such as velocity and pressure. The uncertainties capture both model form uncertainty as well as epistemic uncertainty induced by the limited training data. An invariant Bayesian deep neural network is used to predict the anisotropic tensor component of the Reynolds stress. This model is trained using Stein variational gradient decent algorithm. The computed uncertainty on the Reynolds stress is propagated to the quantities of interest by vanilla Monte Carlo simulation. Results are presented for two test cases that differ geometrically from the training flows at several different Reynolds numbers. The prediction enhancement of the data-driven model is discussed as well as the associated probabilistic bounds for flow properties of interest. Ultimately this framework allows for a quantitative measurement of model confidence and uncertainty quantification for flows in which no high-fidelity observations or prior knowledge is available.

1. Introduction

Over the past decade, with the exponential power increase of computer hardware, computational fluid dynamics (CFD) has become an ever more predominate tool for fluid flow analysis. The Reynolds-averaged Navier–Stokes (RANS) equation provides an efficient method to compute time-averaged turbulent flow quantities making RANS solvers a frequently selected CFD method. However, it is common knowledge that RANS simulations can be highly inaccurate for a variety of flows due to the modeling of the Reynolds stress term [1]. Although over recent years Large Eddy Simulations (LES) or Direct Numerical Simulations (DNS) have become more accessible, these methods still remain out of the scope of practical engineering appli-

* Corresponding author.
E-mail addresses: ngeneva@nd.edu (N. Geneva), nzabaras@gmail.com (N. Zabaras).
URL: https://cics.nd.edu/ (N. Zabaras).

https://doi.org/10.1016/j.jcp.2019.01.021
0021-9991/

cations. For example, design and optimization tasks require repeated simulations with rapid turnaround time requirements for which RANS simulations are the choice modeling tool. Thus improving the accuracy of RANS simulations and providing measures of their predictive capability remains essential for the CFD community.

Turbulence models seek to resolve the closure problem that is brought about from the time averaging of the Navier–Stokes equations. While CFD and computational technology has made significant strides over the past decade, turbulence models have largely become stagnate with the majority of today's most popular models being developed over two decades ago. Many of the most widely used turbulence models employ the Boussinesq assumption as the theoretical foundation combined with a set of parameters that are described through one or more transport equations. In general, these turbulence models can be broken down into families based off the number of additional partial differential equations they introduce into the system. For example, the Spalart–Allmaras model [2] belongs to the family of single equation models. While the Spalart–Allmaras model has been proven to be useful for several aerodynamic related flows [3], its very general structure severely limits the range of flows that it is applicable. In the two-equation family, models such as the $k-\epsilon$ model [4,5] and the $k-\omega$ model [6] provide better modeling for a much larger set of flows even though their limitations are well known. In all the aforementioned models, a set of empirically found constants are used for model-calibration thus resulting in potentially poor performance for flows that were not considered in the calibration process. This combined with empirical modeling of specific transport equations, such as the ϵ equation, result in a significant source of model form uncertainty. While many have proposed more complex approaches such as using different turbulence models for different regions of the flow [7] or using a turbulence model with additional transport equations [8], these methods still rely heavily on empirical tuning and calibration. Thus model form uncertainty introduced by turbulence models continues to be one of the largest sources of uncertainty in RANS simulations.

This work aims to improve turbulence modeling for RANS simulations using machine learning techniques that also allow us to quantify the underlying model error. While the use of machine learning methods in CFD simulations can be traced back to over a decade ago [9], recently there has been a new wave of integrating innovative machine learning algorithms to quantify and improve the accuracy of CFD simulations. Earlier work in quantifying the uncertainty and calibration of turbulence models focused on treating model parameters as random variables and sampling via Monte Carlo to obtain a predictive distribution of outcomes [10,11]. Rather than constraining oneself to a specific model, an alternative approach was to directly perturb components of the anisotropy term of the Reynolds stress [12]. Lately, the use of machine learning models has been shown to provide an efficient alternative to direct sampling. In general, the integration of machine learning with turbulence models can be broken down into three different approaches: modeling the anisotropic term of the Reynolds stress directly, modeling the coefficients of turbulence models and modeling new terms in the turbulence model. Tracey et al. [13] explored the use of kernel regression to model the eigenvalues of the anisotropic term of the Reynolds stress. Later, Tracey et al. [14] used a single layer neural network to predict a source term in the Spalart–Allmaras turbulence model. Similarly, Signh et al. [15] have used neural networks to introduce a functional corrective term to the source term of the Spalart–Allmaras turbulent model for predicting various quantities over airfoils. Zhang et al. [16] investigated the use of neural networks and Gaussian processes to model a correction term introduced to the turbulence model. Ling et al. [17] considered deep neural networks to predict the anisotropic tensor using a neural network structure with embedded invariance [18]. Ling et al. [19] additionally proposed using random forests to improve RANS predictions for a flow with a jet in a cross flow.

While the above works have managed to improve the accuracy of RANS simulations, uncertainty quantification has largely been ignored. Arguably, the integration of black box machine learning models increases the importance of uncertainty quantification in the context of quantifying the error of the improved turbulence model but also quantifying the uncertainty of the machine learning predictions. This is largely due to the significant prediction degradation of these proposed machine learning models for flows that vary from the training data in either fluid properties or geometry [13,17]. Past literature has clearly shown that data-driven methods are not exempt from the conflicting objectives of predictive accuracy versus flow versatility seen in traditional turbulence modeling.

Several works have taken steps towards using machine learning to provide uncertainty quantification analysis of RANS simulations. For example, Xiao et al. [20] proposed a Bayesian data-driven methodology that uses a set of high-fidelity observations to iteratively tune an ensemble of Reynolds-stress fields and other quantities of interest. While proven to work well for even sparse observational data, this work is limited to a single flow with which the machine learning model was trained explicitly on. Wu et al. [21] used the Mahalanobis distance and kernel density estimation to formulate a method to predict the confidence of a data-driven model for a given flow. While this allows the potential identification of regions of less confidence after training, it is limited to the prediction of the anisotropic stress and fails to provide any true probabilistic bounds.

For machine learning methods to be a practical tool for reliably tuning RANS turbulence models, transferability to flows with different geometries and fluid properties is important. Additionally, quantifying the model uncertainty is critical for assessing both the accuracy and confidence of the machine learning model and of the resulting predicted quantities of interest.

The novelty of our work is the use of a data-driven model with a Bayesian deep learning framework to provide the means of improving the accuracy of RANS simulations and allow for the quantification of the model form uncertainty arising in the turbulence model. This uncertainty is then propagated to the quantities of interest, such as pressure and velocity. The focus of our work will not be application on flows that are the same or similar to those in the training set, but rather

to flows defined by different geometries and fluid properties. We aim to take a much more practical and expansive view of using these innovative machine learning models for improved turbulence modeling. The specific novel contributions of this work are fourfold: (a) the use of a Bayesian deep neural network as a model to predict a tuned Reynolds stress field, (b) introducing a stochastic data-driven RANS algorithm that allows us to calculate probabilistic bounds for any flow field quantity, (c) assessment of the data-driven model on flows that are geometrically different from the training simulations and (d) comparison of both performance and confidence of the data-driven model across several Reynolds numbers.

This paper is structured as the following: In Section 2, we review the governing equations and motivation for this work. In Section 3, the proposed data-driven framework is discussed in detail. We discuss the invariant machine learning model in Section 3.1, its extension to the Bayesian paradigm in Section 3.2 and the stochastic data-driven RANS methodology to propagate uncertainty from the Bayesian data-driven model to quantities of interest in Section 3.3. In Section 4, various implementation details are reviewed including information regarding flow data used, training techniques and integration in the selected CFD solver. Section 5 details the results of applying this model to two test flows at three different Reynolds numbers. Results for a flow over a backwards step and over a wall mounted cube are presented in Sections 5.1 and 5.2, respectively. Finally discussion and conclusions are provided in Section 6.

2. Problem formulation

2.1. Governing equations

As previously mentioned, the difficulty of RANS is the fundamental closure problem that is introduced when the Navier–Stokes equations are averaged with respect to time. The RANS momentum equation is as follows:

$$\langle u_j \rangle \frac{\partial \langle u_i \rangle}{\partial x_j} = \frac{\partial}{\partial x_j}\left[-\frac{\langle p \rangle}{\rho}\delta_{ij} + \nu \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right) - \left\langle u_i' u_j' \right\rangle \right] + \langle g_i \rangle . \tag{1}$$

As always, the challenge is to close this equation by approximating the Reynolds stress (R-S) term $\left\langle u_i' u_j' \right\rangle$. u_i' indicates a fluctuation velocity defined as $u_i(x,t) - \langle u_i(x,t) \rangle$ in which $\langle \cdot \rangle$ indicates time-averaged or mean value. The turbulent viscosity theory, originally developed by Boussinesq [22], proposes a form of the R-S that is mathematically analogous to the stress-strain rate of a Newtonian fluid:

$$\left\langle u_i' u_j' \right\rangle = \frac{2}{3}\delta_{ij}k + a_{ij}, \tag{2}$$

$$k = \frac{1}{2}\left\langle u_k' u_k' \right\rangle, \tag{3}$$

$$a_{ij} = -\nu_t \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial \langle u_k \rangle}{\partial x_k} \right), \tag{4}$$

where k, ν_t are the turbulent kinetic energy (TKE) and turbulent viscosity, respectively. Assuming that the flow is incompressible results in the following:

$$\left\langle u_i' u_j' \right\rangle = \frac{2}{3}\delta_{ij}k - \nu_t \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right). \tag{5}$$

This representation is used not because of its accuracy but instead due to the simplifications that result when it is substituted into the RANS equation. This form is known as the *Boussinesq eddy viscosity assumption*.

2.2. RANS turbulence models

The context of this work is focused on the $k-\epsilon$ turbulence model [4,23,24] which is the most commonly used closure model for RANS simulations to date [1]. Starting with the Boussinesq eddy viscosity assumption, the $k-\epsilon$ model approximates the effective viscosity ν_t in terms of the turbulent kinetic energy k and the turbulent dissipation rate ϵ with the R-S given as follows:

$$\left\langle u_i' u_j' \right\rangle = -\tau_{ij} = \frac{2}{3}\delta_{ij}k - \nu_t \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right), \tag{6}$$

$$\nu_t = \frac{C_\mu k^2}{\epsilon}, \tag{7}$$

where C_μ is one of five model constants. Through manipulation of the Navier–Stokes equations, the kinetic energy can be derived precisely for the case of high Reynolds number. On the other hand, the standard transport equation for the

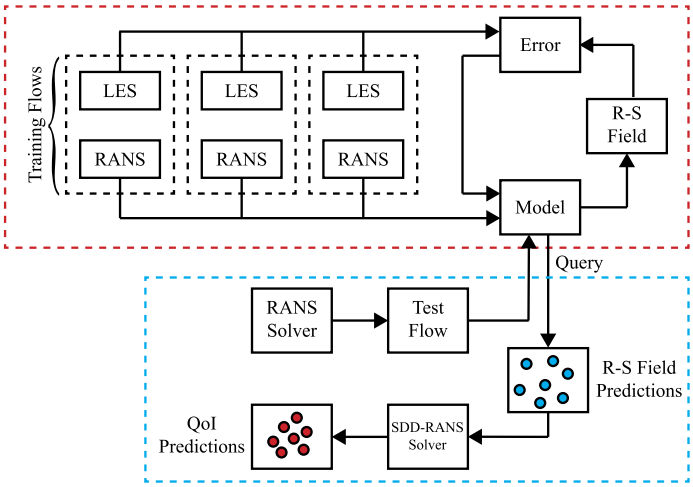

Fig. 1. A schematic of the data-driven Bayesian machine learning framework. The top block illustrates the model training using a set of different flows. Once trained, the model is then queried given a baseline RANS flow and a set of Reynolds stress (R-S) field realizations are sampled. Independent RANS simulations are then performed using these predicted fields by stochastic data-driven RANS (SDD-RANS) and statistics for quantities of interest (QoI) are collected.

turbulent dissipation, ϵ, should be thought of as an empirical fit [1]. For this work, we will use the standard $k-\epsilon$ model for fully-turbulent, incompressible flow [6]:

$$\frac{\partial k}{\partial t} + \langle u_i \rangle \frac{\partial k}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right] + \tau_{ij}\frac{\partial \langle u_i \rangle}{\partial x_j} - \epsilon, \tag{8}$$

$$\frac{\partial \epsilon}{\partial t} + \langle u_i \rangle \frac{\partial \epsilon}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_i}\right] + C_{\epsilon 1}\frac{\epsilon}{k}\tau_{ij}\frac{\partial \langle u_i \rangle}{\partial x_j} - C_{\epsilon 2}\frac{\epsilon^2}{k}. \tag{9}$$

The five constants $C_\mu, C_{\epsilon 1}, C_{\epsilon 2}, \sigma_k, \sigma_\epsilon$ are tunable parameters whose optimal values depend on the flow under consideration. We use the values originally proposed by Launder et al. [23] obtained by data fitting over various turbulent flows:

$$C_\mu = 0.09, \quad C_{\epsilon 1} = 1.44, \quad C_{\epsilon 2} = 1.92, \quad \sigma_k = 1.0, \quad \sigma_\epsilon = 1.3. \tag{10}$$

The advantages of the $k-\epsilon$ model are its numerical robustness, computational efficiency, easy implementation and general validity for fully-turbulent flows. However, with this versatility comes some significant drawbacks including poor accuracy for complex fluid flows, and for problems with flow separation and sharp pressure gradients [7,25]. Core assumptions such as the formulation of the turbulent dissipation equations, the turbulent model constants and even the Boussinesq approximation provide large sources of uncertainty for the $k-\epsilon$ model. Converged simulations using the $k-\epsilon$ model with the parameters discussed above will be referred to as *baseline* RANS simulations. Ultimately, we seek to improve the prediction of a baseline simulation through the proposed data-driven framework.

3. Data-driven framework

In this work, our goal is to introduce a data-driven model to increase the accuracy of a given RANS simulation and to provide uncertainty bounds for quantities of interest thus capturing the error of the turbulence model. The proposed framework is illustrated in Fig. 1 which, in a broad sense, shares similar characteristics to earlier works on data-driven turbulence models [17,20,15]. However, we introduce several novel modifications to the process. We break this framework down into two key phases: the training of a model using a set of pre-existing flow data and the prediction stage for which the model is sampled to produce fluid flow responses.

The training data that is driving our model is a small library of different fluid flows that attempt to capture different fluid physics. Ideally each training flow should bring new information for the model to learn thus increasing its potential predictive capability. For each unique flow, there is a low-fidelity RANS solution and a time-averaged high-fidelity LES solution. The objective of this model is to learn the mapping from some baseline RANS flow input information to a turbulent property yielding an improved R-S field matching that of the corresponding high-fidelity simulation. This turbulent property could be tuned model coefficients, model correction terms or components of the R-S directly. For the scope of this work, we will focus on modeling the R-S tensor directly but this framework can extend to other approaches. An error or loss function that quantifies the discrepancy between the predicted R-S and the true high-fidelity field is used to update the model in an iterative process. We select a Bayesian neural network to serve as this model. Its formulation is discussed in Section 3.1

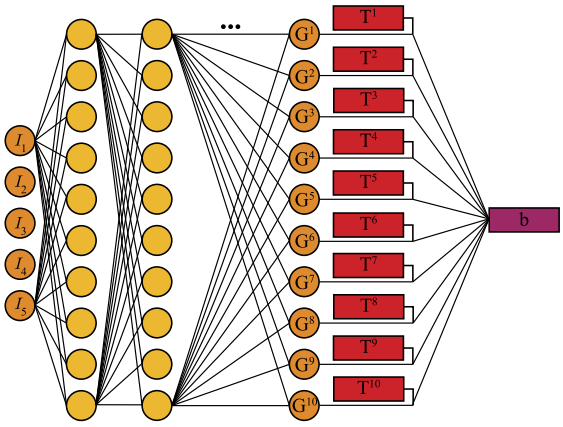

Fig. 2. Invariant, fully-connected (some connections are omitted for clarity), neural network architecture proposed by Ling et al. [17]. The circles indicate scalar values and the rectangles represent 3×3 second-order tensors.

with a Bayesian extension presented in Section 3.2. The methods and techniques used to train the model are outlined in Section 4.

Once the model has been trained, it can be used as a regression model to sample predicted R-S fields for a given reference RANS solution. This process starts with a baseline RANS simulation whose flow field will serve as the input into the calibrated model. From this model, a set of turbulent properties are sampled that correspond to a predicted high-fidelity representation of the R-S field. For each predicted field, an independent forward simulation is completed in which the R-S is held constant and the remaining state variables are relaxed around the predicted field from their baseline values to updated perturbed values. We refer to this process of executing an ensemble of forward simulations as stochastic data-driven RANS (SDD-RANS). The forward simulations for different samples of the R-S can then be used to compute statistical bounds for quantities of interest as discussed in Section 3.3.

Remark 1. LES has been chosen as the high-fidelity method for obtaining the training data in the context of this work. However, one could alternatively use higher accuracy methods such as DNS or even a combination of methods assuming that their turbulent statistics are consistent. The use of LES introduces potential physical inconsistencies in the high-fidelity predictions. Namely, LES can yield different results for the same flow depending on various parameters such as the mesh resolution or the subgrid-scale model used. The use of more consistent DNS data will likely make training more efficient and increase predictive accuracy.

3.1. Invariant neural network

As previously mentioned, in the scope of this work, we will predict the R-S field directly by the anisotropic component shown in Eq. (2). This approach has been used by multiple earlier works [17,20]. These works consider an explicit representation of the R-S component, specifically in the form of a constant field. The remaining fluid flow quantities (mean velocity and pressure) are then *propagated* forward by solving a numerical system around this constrained R-S field. Such an approach does not constrain our work to model-specific assumptions. We note that explicit R-S approaches can potentially result in significant prediction error of fluid quantities when the Reynolds number approaches 5000 and above [26,27]. Additionally, small errors in the predicted R-S field in an explicit representation can be amplified leading to instabilities. We accept this as an open problem and while alternative implicit approaches have been proposed [27], we leave the discussion of such methods to future works. However, we will show how the proposed framework can reflect such difficulties through the predicted probabilistic bounds on the flow quantities of interest.

We select a Bayesian neural network to map the baseline RANS flow to a high-fidelity R-S field due to the impressive performance of neural networks for high-dimensional supervised learning tasks [28]. For the underlying neural network model, we choose the neural network proposed by Ling et al. [17], illustrated in Fig. 2. This neural network predicts the anisotropic tensor of the R-S using the symmetric and antisymmetric tensor components of the velocity gradient tensor. Through use of tensor invariants, the neural network is able to achieve both Galilean invariance as well as invariance to coordinate transformations. This makes such a model attractive for predictions of flows that deviate in geometry from the training data. Here, we briefly review the fundamentals of this neural network for completeness of the presentation.

The theoretical foundation of this invariant neural network is the non-linear eddy viscosity model developed by Pope [29]. In this model, the normalized anisotropic tensor of the R-S is expressed as a function, $\boldsymbol{b}(\boldsymbol{s}, \boldsymbol{\omega})$, of the normalized mean rate-of-strain tensor $\boldsymbol{s}$ and rotation tensor $\boldsymbol{\omega}$:

$$\left\langle u_i' u_j' \right\rangle = \frac{2}{3}\delta_{ij}k + kb_{ij}(\boldsymbol{s}, \boldsymbol{\omega}), \tag{11}$$

$$s_{ij} = \frac{1}{2}\frac{k}{\epsilon}\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i}\right), \quad \omega_{ij} = \frac{1}{2}\frac{k}{\epsilon}\left(\frac{\partial \langle u_i \rangle}{\partial x_j} - \frac{\partial \langle u_j \rangle}{\partial x_i}\right), \tag{12}$$

where both $\boldsymbol{s}$ and $\boldsymbol{\omega}$ are scaled by the TKE and turbulent dissipation. For clarity we will refer to the tensor, $\boldsymbol{a} = k \cdot \boldsymbol{b}(\boldsymbol{s}, \boldsymbol{\omega})$, used when solving the RANS equations as the unnormalized anisotropic tensor. Through application of the Cayley–Hamilton theorem, it can be shown that every second-order anisotropic tensor can be expressed in the following form:

$$\boldsymbol{b}(\boldsymbol{s}, \boldsymbol{\omega}) = \sum_{\lambda=1}^{10} G^{\lambda}\left(\mathcal{I}_{1:5}\right)\boldsymbol{T}^{\lambda}, \tag{13}$$

$$\mathcal{I}_i = \left\{Tr(\boldsymbol{s}^2),\ Tr(\boldsymbol{\omega}^2),\ Tr(\boldsymbol{s}^3),\ Tr(\boldsymbol{\omega}^2\boldsymbol{s}),\ Tr(\boldsymbol{\omega}^2\boldsymbol{s}^2)\right\}, \tag{14}$$

$$\begin{aligned}
&\boldsymbol{T}^1 = \boldsymbol{s}, && \boldsymbol{T}^2 = \boldsymbol{s}\boldsymbol{\omega} - \boldsymbol{\omega}\boldsymbol{s}, && \boldsymbol{T}^3 = \boldsymbol{s}^2 - \frac{1}{3}\boldsymbol{I}\mathrm{Tr}\left(\boldsymbol{s}^2\right),\\
&\boldsymbol{T}^4 = \boldsymbol{\omega}^2 - \frac{1}{3}\boldsymbol{I}\mathrm{Tr}\left(\boldsymbol{\omega}^2\right), && \boldsymbol{T}^5 = \boldsymbol{\omega}\boldsymbol{s}^2 - \boldsymbol{s}^2\boldsymbol{\omega}, && \boldsymbol{T}^6 = \boldsymbol{\omega}^2\boldsymbol{s} + \boldsymbol{s}\boldsymbol{\omega}^2 - \frac{2}{3}\boldsymbol{I}\mathrm{Tr}(\boldsymbol{s}\boldsymbol{\omega}^2),\\
&\boldsymbol{T}^7 = \boldsymbol{\omega}\boldsymbol{s}\boldsymbol{\omega}^2 - \boldsymbol{\omega}^2\boldsymbol{s}\boldsymbol{\omega}, && \boldsymbol{T}^8 = \boldsymbol{s}\boldsymbol{\omega}\boldsymbol{s}^2 - \boldsymbol{s}^2\boldsymbol{\omega}\boldsymbol{s}, && \boldsymbol{T}^9 = \boldsymbol{\omega}^2\boldsymbol{s}^2 + \boldsymbol{s}^2\boldsymbol{\omega}^2 - \frac{2}{3}\boldsymbol{I}\mathrm{Tr}(\boldsymbol{s}^2\boldsymbol{\omega}^2),\\
&\boldsymbol{T}^{10} = \boldsymbol{\omega}\boldsymbol{s}^2\boldsymbol{\omega}^2 - \boldsymbol{\omega}^2\boldsymbol{s}^2\boldsymbol{\omega},
\end{aligned} \tag{15}$$

where $\boldsymbol{T}^{\lambda}$ is one of 10 independent, symmetric tensor functions and G^{λ} are the respective coefficients in the linear model which can be each expressed as functions of the five invariants $\mathcal{I}_1, \cdots, \mathcal{I}_5$. For complete details on the invariants, tensor functions and the derivation of the representation above, we refer the reader to [29].

The neural network model proposed by Ling et al. [17] models the anisotropic term by using the linear combination in Eq. (13). As illustrated in Fig. 2, rather than using the components of the symmetric and antisymmetric tensors ($\boldsymbol{s}$ and $\boldsymbol{\omega}$) directly, the invariants and tensor basis functions in Eqs. (14) and (15) are used instead. To enforce invariance to coordinate transformations, the neural network is used to learn the tensor basis coefficients G^{λ} which are functions of the five invariants in Eq. (14). These predicted coefficients, G^{λ}, can then be used with the tensor basis functions, $\boldsymbol{T}^{\lambda}$, to produce the anisotropic tensor $\boldsymbol{b}$. Thus while the model predicts the anisotropic tensor given the symmetric and antisymmetric tensors of the velocity gradient, the basis coefficients as functions of the five invariants is what is being learned. If a model uses inputs with specific invariant properties, the model has the same invariance properties as well [30]. This allows the neural network to be (a) Galilean invariant due to the use of the rate-of-strain and rotation tensors which are functions of the velocity gradient; and (b) invariant to coordinate transformations through the use of the invariant inputs $\mathcal{I}_i$. Additionally, since this eddy viscosity model is the most general formulation, this neural network model does not share any of the limitations of other simpler models that place restrictions on the form of the anisotropic term. However, an intrinsic assumption of this model is that the mapping between the RANS and LES physical domains can be thoroughly expressed by the invariants $\mathcal{I}_i$. This is clearly not guaranteed, however, the introduction of additional input features would potentially result in loss of coordinate system invariance thus degrading model generalization.

Remark 2. This neural network formulation is trained on entirely local (point-wise) information. The key advantage of a spatially local model is that it extends very easily to training flow data provided on non-uniform meshes which are essential in practical CFD simulations. Approaches such as convolution neural networks require training data on a uniform mesh following an image-to-image like regression approach [31]. However, similar to turbulent eddy viscosity models, this approach implies that the R-S mean convection $D\left\langle u_i' u_j' \right\rangle / Dt$ is governed entirely by local quantities (*e.g.* k, ϵ, $\partial \langle u_i \rangle / \partial x_j$). This is a questionable assumption for flows that exhibit strong inhomogeneity [1]. A model that incorporates spatial correlations would likely be more descriptive, physically robust and potentially easier to train.

3.2. Bayesian neural network

Traditionally neural networks are not designed to yield predictive statistics, however multiple recent works explore Bayesian reformulations of neural networks. Older techniques for obtaining Bayesian statistics include the placement of distributions over network weights and sampling with Monte Carlo methods to approximate statistical bounds [32,33] as well as ensemble methods [34,35]. More recently, methods involving stochastic variational inference have brought a new wave a Bayesian neural network techniques [36–39]. In this work, we choose to use Stein variational gradient decent (SVGD) recently proposed by Liu et al. [39,40] that approximates a variational distribution through a set of particles. SVGD is a non-parametric algorithm of similar form as standard gradient decent.

We follow closely the work of Zhu and Zabaras [31] in which SVGD is successfully applied to deep convolutional neural networks used for surrogate modeling. For the invariant neural network architecture discussed previously, we will use the following representation:

$$\boldsymbol{b} = \boldsymbol{f}(\{\boldsymbol{s}, \boldsymbol{\omega}\}, \mathbf{w}) = \boldsymbol{f}(\boldsymbol{x}, \mathbf{w}), \tag{16}$$

where the input $\boldsymbol{x} = \{\boldsymbol{s}, \boldsymbol{\omega}\}$ consists of the strain and rotation tensors $\boldsymbol{s}$ and $\boldsymbol{\omega}$ along with the neural networks parameters $\mathbf{w}$ which include weights and biases. For mathematical convenience, we will represent the anisotropic tensor with a one-dimensional vector $\boldsymbol{b} \in \mathbb{R}^9$ for the remainder of this section. In the equation above, we have defined the neural network model as a function that has absorbed the calculation of the invariants, tensor basis functions and the linear combination detailed in Eqs. (13)–(15). Thus we will refer to the function $\boldsymbol{f}$ as the invariant neural network model. We wish to treat the neural network's K learnable parameters as random variables. Due to the potentially large number of weights in a fully-connected neural network, we assume that the weights have a probability density function of a fully-factorizable zero mean Gaussian and Gamma-distributed precision scalar α:

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\boldsymbol{I}_K), \quad p(\alpha) = Gamma(\alpha|a_0, b_0), \tag{17}$$

where the rate a_0 and shape parameters b_0 are taken as 1.0 and 0.025, respectively and $\boldsymbol{I}_n$ denotes the identity matrix in $\mathbb{R}^{n\times n}$. The resulting prior has the density of a narrow Student's $\mathcal{T}$-distribution centered at zero. This promotes sparsity [41] and helps to prevent over-fitting. With the use of a sufficient number of weights and the highly non-linear nature of the neural network, such a prior places little restriction on the network's final functional form [33]. Additionally, output-wise noise is added onto the predicted output to represent inherent uncertainty within the model's formulation or uncertainty that cannot be reduced with more training data. This results in an additional noise term to the likelihood function of the neural network. We assume that the noise takes the form of a zero mean Gaussian with a learnable precision β that is Gamma distributed:

$$\boldsymbol{b} = \boldsymbol{f}(\boldsymbol{x}, \mathbf{w}) + \boldsymbol{\epsilon}, \tag{18}$$

$$p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|0, \beta^{-1}\boldsymbol{I}_9), \quad p(\boldsymbol{b}) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{f}(\boldsymbol{x}, \mathbf{w}), \boldsymbol{I}_9), \tag{19}$$

$$p(\beta) = Gamma(\beta|a_1, b_1). \tag{20}$$

Since both the LES and RANS simulations are being used for the same flows, we assume that the LES solution will be statistically stationary. We also assume that the LES data has sufficiently converged by averaging over an adequate number of time steps. The output-size noise is assumed to have a small variance and thus we assign in the prior for β the shape and rate parameters to be $a_1 = 100$ and $b_1 = 2 \cdot 10^{-4}$, respectively. This weakly promotes large β with an expected value of 5×10^5 and a variance on the order of 10^{-3}, which is less than one percent of the scaled training data range. For the sake of brevity, we will drop the notation of the conditional dependence on the hyper-parameters a_0, b_0, a_1 and b_1 implying that the posterior distribution will be conditionally dependent on these terms. To optimize the parameters in the neural network, SVGD minimizes the Kullback–Leibler (KL) divergence between the true parameter posterior, $p(\mathbf{w}, \beta|\mathcal{D})$, given the batch of M i.i.d. training data $\mathcal{D} = \{\boldsymbol{b}_i\}_{i=1}^M$, with the variational distribution $q(\mathbf{w}, \beta)$ that lies in some set of distributions $\mathcal{Q}$:

$$q^*(\mathbf{w}, \beta) = \min_{q\in\mathcal{Q}} \left\{ \mathrm{KL}(q||p) \equiv \mathbb{E}_q(\log q(\mathbf{w}, \beta)) - \mathbb{E}_q(\log \widetilde{p}(\mathbf{w}, \beta|\mathcal{D})) + \mathcal{K} \right\}, \tag{21}$$

for which $\widetilde{p}(\mathbf{w}, \beta|\mathcal{D})$ is the unnormalized posterior and $\mathcal{K}$ is the log normalization constant that is not required to be computed during optimization. For the given neural network, we prescribe a Gaussian likelihood function and the priors discussed previously:

$$\widetilde{p}(\mathbf{w}, \beta|\mathcal{D}) = p(\mathcal{D}|\mathbf{w}, \beta)p(\mathbf{w}, \beta), \tag{22}$$

$$\widetilde{p}(\mathbf{w}, \beta|\mathcal{D}) = \prod_{i=1}^{M} \left[\mathcal{N}(\boldsymbol{b}_i|\boldsymbol{f}(\boldsymbol{x}_i, \mathbf{w}), \beta^{-1}\boldsymbol{I}_9)\right] \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\boldsymbol{I}_K)\Gamma(\alpha|a_0, b_0)\Gamma(\beta|a_1, b_1). \tag{23}$$

Rather than attempting to recover a parametric form of the variational distribution, SVGD describes $q(\mathbf{w}, \beta)$ by a particle approximation. Namely, a set of N deterministic neural networks each representing a particle $\{\boldsymbol{\theta}_i\}_{i=1}^N$, $\boldsymbol{\theta}_i = \{\mathbf{w}_i, \beta_i\}$, leading to an empirical measure $q_N(\mathbf{w}', \beta') = q_N(\boldsymbol{\theta}') = \frac{1}{N}\sum_{i=1}^N \delta(\boldsymbol{\theta}_i - \boldsymbol{\theta}')$. Thus the objective is now for the empirical probability measure, μ_N, to converge in distribution towards the true measure of the posterior ν,

$$\mu_N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \delta(\boldsymbol{\theta}_i - \boldsymbol{\theta}) d\boldsymbol{\theta} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\theta}_i, \tag{24}$$

$$\nu(d\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \tag{25}$$

To minimize the KL divergence, we assume that $q(\mathbf{w}, \beta)$ is from a class of distributions that can be obtained through a set of smooth transforms. A small perturbation function, resembling that of standard gradient decent, is used to update the particles:

$$\boldsymbol{\theta}_i^{t+1} = \boldsymbol{T}(\boldsymbol{\theta}_i^t) = \boldsymbol{\theta}_i^t + \eta^t \boldsymbol{\phi}(\boldsymbol{\theta}_i^t), \tag{26}$$

where η is the step size and $\boldsymbol{\phi}(\boldsymbol{\theta}_i^t)$ is the direction of the update that lies in a function space $\mathcal{F}$ for the t-th iteration. It is now a matter of finding the optimal direction to permute the particles which should be chosen such that the KL divergence is maximally reduced, namely,

$$\boldsymbol{\phi}^* = \max_{\boldsymbol{\phi} \in \mathcal{F}} \left(-\frac{d}{d\eta} \mathcal{KL}(\boldsymbol{T}\mu_N || \nu)|_{\eta=0} \right), \tag{27}$$

where $\boldsymbol{T}\mu$ denotes the updated empirical measure of the particles. Liu et al. [39] identify connections between the function $\boldsymbol{\phi}$ and Stein's method and show that:

$$\frac{\partial}{\partial \eta} KL(\boldsymbol{T}\mu_N || \nu)|_{\eta=0} = \mathbb{E}_\mu \left(\mathcal{T}_p \boldsymbol{\phi} \right), \quad \mathcal{T}_p \boldsymbol{\phi} = (\nabla \log p(\boldsymbol{\theta}|\mathcal{D})) \cdot \boldsymbol{\phi} + \nabla \cdot \boldsymbol{\phi}, \tag{28}$$

in which $\mathcal{T}_p$ is known as the Stein's operator. Assuming that this function space $\mathcal{F}$ is a unit ball in a reproducing kernel Hilbert space $\mathcal{H}$ with positive kernel $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$, the optimal direction has the closed form:

$$\boldsymbol{\phi}^*(\boldsymbol{\theta}) \propto \mathbb{E}_{\boldsymbol{\theta}' \sim \mu} \left[\left(\nabla_{\boldsymbol{\theta}'} \log p(\boldsymbol{\theta}'|\mathcal{D}) \right) k(\boldsymbol{\theta}, \boldsymbol{\theta}') + \nabla_{\boldsymbol{\theta}'} k(\boldsymbol{\theta}, \boldsymbol{\theta}') \right], \tag{29}$$

where $p(\boldsymbol{\theta}'|\mathcal{D})$ is given by Eq. (23). In this work, we choose to use the standard radial basis function kernel for $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$. This formulation results in a simple update procedure in which the optimal decent direction for all particles is calculated with Eq. (29) and then updated by Eq. (26). Monte Carlo approximations can then be used to find the predictive mean:

$$\begin{aligned} \mathbb{E}(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathcal{D}) &= \mathbb{E}_{p(\mathbf{w},\beta|\mathcal{D})}(\mathbb{E}(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathbf{w}, \beta)) \\ &= \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}(\boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w})) \approx \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w}_i), \end{aligned} \tag{30}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{b}^*$ are the test input and corresponding predictive model output, respectively. The output noise is not present due to its density of a zero mean Gaussian. The approximation of the predictive variance similarly follows:

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathcal{D}) &= \mathbb{E}_{p(\mathbf{w},\beta|\mathcal{D})}(\mathrm{Cov}(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathbf{w}, \beta)) + \mathrm{Cov}_{p(\mathbf{w},\beta|\mathcal{D})}(\mathbb{E}(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathbf{w}, \beta)) \\ &= \mathbb{E}_{p(\beta|\mathcal{D})}(\beta^{-1}\boldsymbol{I}_9) + \mathrm{Cov}_{p(\mathbf{w}|\mathcal{D})}(\boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w})) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \left((\beta_i)^{-1} \boldsymbol{I}_9 + \boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w}_i) \boldsymbol{f}^T(\boldsymbol{x}^*, \mathbf{w}_i) \right) \\ &\quad - \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}(\boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w})) \mathbb{E}^T_{p(\mathbf{w}|\mathcal{D})}(\boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w})), \end{aligned} \tag{31}$$

in which $\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}(\boldsymbol{f}(\boldsymbol{x}^*, \mathbf{w}))$ is calculated in Eq. (30). Although the focus of this paper is to investigate the effect of the model form uncertainty on fluid quantities, the Bayesian neural network also allows for rigorous study of the epistemic uncertainty with Eqs. (30) and (31). Thus one can study the effect of training data, model architecture, and other parameters on predictive confidence. For complete details on SVGD, we direct the reader to the original work by Liu et al. [39,40] along with the work of Zhu and Zabaras [31].

3.3. Uncertainty quantification with SDD-RANS

We now wish to propagate this uncertainty obtained for the anisotropic term to the fluid properties such as pressure or velocity. We use a stochastic system approach for which the model parameters in a system of PDEs are considered as random variables. This methodology has been used extensively in the past for model calibration, prediction and selection [42–44,10]. Consider a dynamical system defined by the model output $h(\boldsymbol{\phi}, \boldsymbol{u}(\boldsymbol{\phi}))$, where $\boldsymbol{u}(\boldsymbol{\phi})$ are state variables that evolve with the dynamical system and $\boldsymbol{\phi}$ represents a set of model parameters with probability density $p(\boldsymbol{\phi})$. Traditionally the true form of the distribution $p(\boldsymbol{\phi})$ from which the model parameters are sampled from is largely not known. However, under the assumption that samples can be drawn from the parameter distribution, the expected response as well as the respective variance can be approximated by vanilla Monte Carlo simulation (MCS) with P samples of the random model parameters:

$$\mathbb{E}_{p(\boldsymbol{\phi})}(h) \approx \frac{1}{P}\sum_{i=1}^{P} h\left(\boldsymbol{\phi}_i, \boldsymbol{u}(\boldsymbol{\phi}_i)\right), \quad \boldsymbol{\phi}_i \sim p(\boldsymbol{\phi}), \tag{32}$$

$$\mathrm{Var}_{p(\boldsymbol{\phi})}(h) \approx \frac{1}{P}\sum_{i=1}^{P}\left[h\left(\boldsymbol{\phi}_i, \boldsymbol{u}(\boldsymbol{\phi}_i)\right) - \mathbb{E}_{p(\boldsymbol{\phi})}(h)\right]^2. \tag{33}$$

To extend this to the problem of interest and motivate SDD-RANS, let us consider the model output h as the flow field predicted by the RANS equations and the state variables $\boldsymbol{u}(\boldsymbol{\phi})$ to be the fluid's velocity, pressure and all other derived properties. As previously discussed, we will be taking an explicit representation of the tuned R-S in which a modified R-S field is predicted and held constant while the other state variables are propagated forward. Thus we are able to view a predicted R-S field as a random model parameter, namely, $p(\boldsymbol{\phi}) = p(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathcal{D})$ where the predictive density of $\boldsymbol{b}^*$ is given by:

$$p(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{b}^*|\boldsymbol{x}^*, \mathbf{w}, \beta)p(\mathbf{w}, \beta|\mathcal{D})d\mathbf{w}d\beta. \tag{34}$$

Rather than sampling the anisotropic term directly from the predictive distribution, recall the following representation of the likelihood in Eq. (18):

$$\boldsymbol{b}^* = \boldsymbol{f}\left(\left\{\boldsymbol{s}^*, \boldsymbol{\omega}^*\right\}, \mathbf{w}\right) + \boldsymbol{\epsilon}.$$

To sample the predictive distribution, one can first sample the posterior $p(\mathbf{w}, \beta|\mathcal{D})$ and then execute a forward prediction of the neural network as well as sample the additive output noise yielding the predicted $\boldsymbol{b}^*$. We can modify the MCS such that we sample the weights of the Bayesian neural network as well as the variance of the additive output-wise noise:

$$\mathbb{E}_{p(\mathbf{w},\beta|\mathcal{D})}(h) \approx \frac{1}{N}\sum_{i=1}^{N} h\left(\boldsymbol{b}_i^*, \boldsymbol{u}(\boldsymbol{b}_i^*)\right), \tag{35}$$

$$\mathrm{Var}_{p(\mathbf{w},\beta|\mathcal{D})}(h) \approx \frac{1}{N}\sum_{i=1}^{N}\left[h\left(\boldsymbol{b}_i^*, \boldsymbol{u}(\boldsymbol{b}_i^*)\right) - \mathbb{E}_{p(\mathbf{w},\beta|\mathcal{D})}(h)\right]^2, \tag{36}$$

$$\boldsymbol{b}_i^* = \boldsymbol{f}\left(\left\{\boldsymbol{s}^*, \boldsymbol{\omega}^*\right\}, \mathbf{w}_i\right) + \boldsymbol{\epsilon}_i, \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{\epsilon}_i|0, \beta_i^{-1}\boldsymbol{I}_9), \tag{37}$$

$$\{\mathbf{w}_i, \beta_i\} \sim p(\mathbf{w}_i, \beta_i|\mathcal{D}). \tag{38}$$

The SVGD algorithm provides samples of the posterior $p(\mathbf{w}_i, \beta_i|\mathcal{D})$. Namely, given that SVGD uses a particle representation, each sample is a particle (or invariant neural network) used during training. In practice, the output-wise noise, $\boldsymbol{\epsilon}$, has minimal influence on the predicted values due to the previously made assumptions. Hence, we only take a mean point estimate of the likelihood. In principle, the neural network's inputs are spatially independent between mesh nodes allowing for each node point in the fluid domain to have independent weight samples resulting in a stochastic field. However, the use of the divergence of the R-S in the RANS equations suggests that the spacial smoothness of the predicted field is of significant importance. Thus, we use a single neural network, $\boldsymbol{f}(\{\boldsymbol{s}^*, \boldsymbol{\omega}^*\}, \mathbf{w}_i)$, to predict the R-S for the entire flow domain. As a result, in the context of the fluid domain, we are in fact sampling a functional representation of the R-S that is dependent on the velocity gradients. This combination of using a Bayesian data-driven model with a stochastic model parameter is why we have named this process stochastic data-driven RANS (SDD-RANS). With SDD-RANS, we have opened up the ability to obtain sample statistics for all flow quantities through traditional MCS. This allows for the quantification of uncertainty regarding our data-driven model beyond the R-S itself.

The use of the explicit representation of the R-S and the noisy nature of the neural network's predictions raise concerns regarding the convergence of the SDD-RANS model. In practice, at higher Reynolds numbers the simulation may fail to converge in some areas of the domain. Due to the nature of SDD-RANS, the statistical averages obtained through MCS accurately reflect the true state of the quantities of interest. The discrepancy from the true solution is reflected by the computed variance or uncertainty estimates. However, while computing each sample, one must still monitor the residuals of the model to ensure the initial transient state has ended. In practice, we run the forward simulation for the same number of iterations as the baseline simulation.

3.4. Framework implementation

We use this Bayesian framework in the system of RANS equations by setting the R-S term as the stochastic parameter that is sampled from the predictive distribution obtained through the Bayesian neural network. We summarize the offline training process:

- The training data consist of both baseline RANS and high-fidelity data for a set of different flows that attempt to capture different flow physics.

Table 1
Mesh and CFD parameters for each training flow which includes the respective reference, mesh sizes for both RANS and LES simulations, the domain size, characteristic length L_c, bulk Reynolds number Re_b and kinematic viscosity ν. Streamwise is in the x-, wall normal in the y- and spanwise in the z-direction.

Case	Converge diverge	Square cylinder	Periodic hills	Square duct	Tandem cylinders
Reference	Schiavo et al. [50,51]	Bosch et al. [52]	Temmerman et al. [53,54]	Pinelli et al. [55]	Gopalan et al. [56]
Mesh RANS	$140 \times 50 \times 50$	$100 \times 60 \times 20$	$100 \times 50 \times 50$	$7.5\pi \times 60 \times 60$	$80 \times 60 \times 50$
Mesh LES	$280 \times 100 \times 150$	$280 \times 120 \times 40$	$500 \times 150 \times 250$	$300\pi \times 150 \times 150$	$470 \times 180 \times 120$
Domain Size	$12.56H \times 2H \times 3H$	$20D \times 14D \times 4D$	$9H \times 3.306H \times 4.5H$	$4\pi H \times 2H \times 2H$	$30D \times 20D \times 3D$
L_c	Half Channel Height	Cylinder Diameter	Hill Height	Half Channel Width	Cylinder Diameter
Re_b	5000	5000	6210	6680	5000
ν	2.00e−4	2.00e−4	6.07e−4	2.00e−4	7.40e−4

Table 2
Mesh and CFD parameters for each test flow which includes the respective reference, mesh sizes for both RANS and LES simulations, the domain size, characteristic length L_c, bulk Reynolds number Re_b and kinematic viscosity ν. Streamwise is in the x-, wall normal in the y- and spanwise in the z-direction.

Case	Backward step	Wall mounted cube
Reference	Gresho et al. [57]	Yakhot et al. [58]
Mesh RANS	$220 \times 60 \times 20$	$100 \times 40 \times 80$
Mesh LES	$390 \times 100 \times 40$	$200 \times 100 \times 150$
Domain Size	$27H \times 2H \times H$	$14H \times 3H \times 7H$
L_c	Step Height	Cube Height
Re_b	500, 2500, 5000	500, 2500, 5000
ν	2.00e−3, 4.00e−4, 2.00e−4	2.00e−3, 4.00e−4, 2.00e−4

- The underlying machine learning model is an invariant neural network that uses local fluid quantities to predict the anisotropic term of the R-S.
- We extend this invariant model to the Bayesian paradigm by using SVGD in which a set of neural networks approximates the posterior $p(\mathbf{w}, \beta|\mathcal{D})$ by a particle representation.
- The parameters in each particle (or neural network) are optimized by minimizing the KL divergence between a particle variational approximation and the posterior of the parameters.
- An iterative algorithm, resembling the form of standard gradient decent, updates the parameters of each particle until convergence.

With the Bayesian neural network trained, one can make predictions for new flows:

- For the flow of interest, a baseline RANS solution is obtained and the corresponding invariants and tensor functions at each mesh point are calculated.
- Each neural network used during training with SVGD is used to predict a corresponding high-fidelity R-S field.
- For each predicted field, the R-S is then constrained to the predicted values and a forward execution of the constrained system updates the remaining state variables.
- An equivalent number of state variable samples are then obtained for which probabilistic bounds can be calculated.

4. Numerical implementation and training

4.1. CFD methods

For obtaining the training and test flows, the open source CFD platform OpenFOAM (Open source Field Operation And Manipulation) [45,46] is used. OpenFOAM is a widely accepted CFD package that contains a vast number of solvers for incompressible, compressible and multi-phase flows along with pre- and post-processing utilities. For the baseline RANS simulations the steady-state, incompressible solver *simpleFoam* was used which employs the semi-implicit method for pressure linked equations (SIMPLE) algorithm [47] to solve both the momentum and pressure equations. The high-fidelity LES simulations used the *pimpleFoam* transient solver that combines both the PISO (Pressure Implicit with Split Operator) [48] and SIMPLE algorithms to solve the pressure and momentum equations. The Smagorinsky subgrid-scale model [49] with Van-Driest style damping was used for all LES flows. Both the baseline RANS and LES domains are discretized by second-order methods. Each training and testing flow is outlined in Tables 1 and 2, and all meshes are non-uniform such that the mesh density increases around the feature of interest. All simulations were run with a CFL number below 0.3 for numerical accuracy.

Table 3
Neural network architecture and training details.

Architecture	$5 \to 200 \to 200 \to 200 \to 40 \to 20 \to 10$
Activation	Leaky ReLu
Optimizer	ADAM [60]
Weight Decay	0.01
Learning Rate	5e−6, with learning rate decay on plateau
Epochs	100
Training Data	10000
Mini-batch size	20
SVGD Particles	20

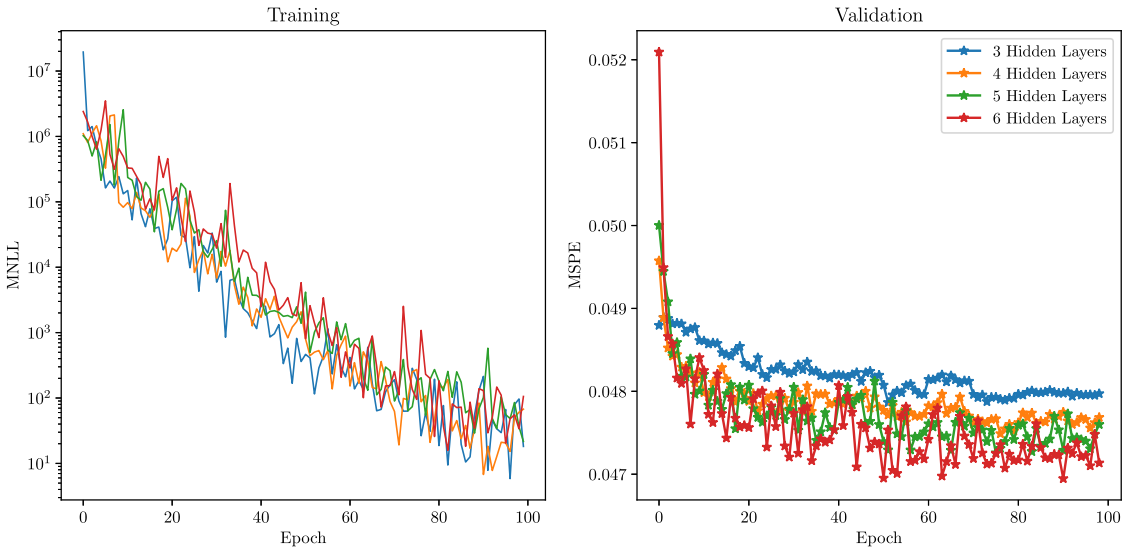

Fig. 3. (Left) The negative log likelihood (MNLL) and (Right) the mean squared prediction error (MSPE) of the validation data for several neural network architectures. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4.2. Machine learning implementation

To train the neural network, the Python machine learning library PyTorch [59] was used. The software and data used in this work are available at https://github.com/cics-nd/rans-uncertainty. The details of the network architecture used are given in Table 3. The Leaky Rectifier function was used as the activation function as opposed to the standard Rectifier function to prevent too many nodes from becoming zero during training. Additionally the number of nodes in the hidden layers is tapered at the end of the network to prevent weights from being too small, which improved training performance.

The network architecture was determined by training an ensemble of neural networks with different number of hidden layers. Other network parameters, such as the taper of the last several layers and learning rate specified in Table 3, were identical between each of the tested architectures. The networks are compared in Fig. 3 with the mean negative log likelihood (MNLL) defined by:

$$MNLL = -\frac{1}{T}\sum_{i=1}^{T}\frac{1}{N}\sum_{j=1}^{N}\log p(\hat{\boldsymbol{b}}_i|\boldsymbol{f}(\hat{\boldsymbol{x}}_i,\mathbf{w}_j),\beta_j), \tag{39}$$

where T, $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{b}}$ are the number of validation/test data points, the target inputs and target outputs, respectively. We observe little distinguishable difference indicating that training between each architecture is relatively the same. Additionally, the mean squared prediction error of the unnormalized anisotropic tensor $\boldsymbol{a}^*$ is plotted for a validation set of 1000 random data points from each of the training flows (i.e. 5000 total data points). The mean squared prediction error (MSPE) is defined as:

$$MSPE = \frac{1}{T}\sum_{i=1}^{T}\left\|\mathbb{E}(\boldsymbol{a}_i^*|\hat{\boldsymbol{x}}_i,\mathcal{D}) - \hat{\boldsymbol{a}}_i\right\|_2^2 \approx \frac{1}{T}\sum_{i=1}^{T}\left\|\frac{k}{N}\sum_{j=1}^{N}\boldsymbol{f}(\hat{\boldsymbol{x}}_i,\mathbf{w}_i) - \hat{\boldsymbol{a}}_j\right\|_2^2, \tag{40}$$

where k is the baseline RANS TKE and $\hat{\boldsymbol{a}}$ are the target unnormalized anisotropic tensors. For the MSPE, there is a notable difference between the converged accuracy of each network. The networks with above 6 hidden layers exhibited significant

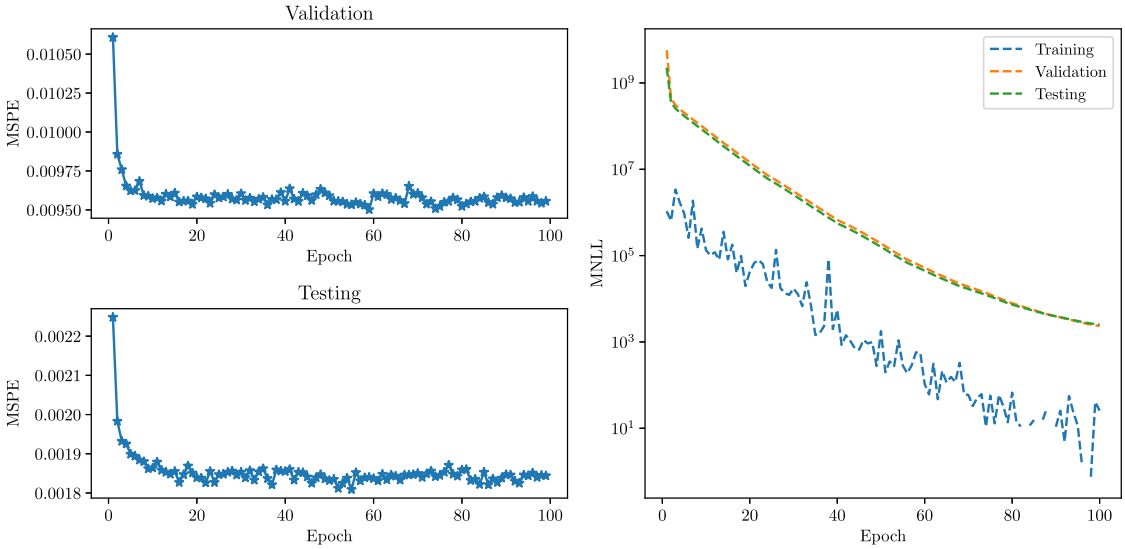

Fig. 4. (Left) The mean squared prediction error (MSPE) of both the validation and test datasets. (Right) The mean negative log likelihood (MNLL) of the training, validation and test datasets.

over-fitting of the validation data and are not plotted. One can observe the onset of over-fitting by the noisy MSPE of the 6 hidden layer neural network. The network architecture with 5 hidden layers was selected to ensure that over-fitting does not take place.

Compared to other potential network models, we found that the model selected originally by Ling et al. [17] proved to be exceptionally difficult to train. This is reflected in the original work by the extremely low learning rate used of 2.5×10^{-6}. During training we also found only very low learning rate could be used for the training process to be stable. To increase training performance and efficiency, we also used the following techniques:

- Depending on the size of the fluid domains, the number of training points for a single flow can be large. Thus to increase training efficiency, rather than using every single mesh point, a subset of training points is selected. In this work, we use only 10^4 total training points that are evenly distributed among all training flows (i.e. 2×10^3 points for each of the five test flows). Every 10 epochs these points are then re-sampled at random. This prevents the potential issue of exceeding the available memory on the provided GPU.
- Training points are shuffled randomly and mini-batched every epoch such that data from multiple flows can reside in a single mini-batch. This helps prevent the model from over-fitting to a specific flow and improves the quality of predictions.
- The invariant inputs to the neural network tended to vary strongly in magnitude including very large values near fixed boundaries. Thus the invariants are re-scaled by a sigmoidal operation that helps to normalize outliers to the range of $+1$ to -1 [61]. In addition, the tensor basis functions were normalized by the L_2 norm of the matrix:

$$\hat{\mathcal{I}}_i = \frac{1 - e^{-\mathcal{I}_i}}{1 + e^{-\mathcal{I}_i}}, \qquad \hat{\boldsymbol{T}}^\lambda = \frac{\boldsymbol{T}^\lambda}{\left\| \boldsymbol{T}^\lambda \right\|_2}. \tag{41}$$

To quantify the training quality, the MSPE is calculated for both the validation set along with a test set of 1000 randomly selected points from each test flow in Table 2. Additionally, we also plot the MNLL for the training, validation and testing data sets. The results are illustrated in Fig. 4. We note that for both the validation and test datasets the model quickly converges and exhibits minimal over-fitting. The training process on a single NVIDIA P100 GPU took approximately 3.0 wall-clock hours.

To verify that the trained model has learned a physical interpretation of the training data, we plot the contours of the mixing coefficients G^λ predicted by the neural network for the square cylinder and periodic hills training flows in Figs. 5 and 6, respectively. While there appears to be some minor over-fitting in front of the square cylinder in Fig. 5, for both flows it is clear that the model has indeed identified physical regions of the flow as well as maintained symmetries.

4.3. Constrained R-S simulation

To integrate the sampled R-S field into OpenFOAM, a small modification is made to the *simpleFoam* solver such that the R-S is now a constant field in the momentum RANS equation. Since the R-S field is held constant, the calculation of the TKE and turbulent dissipation is no longer needed. An important issue is the handling of boundary conditions. This includes

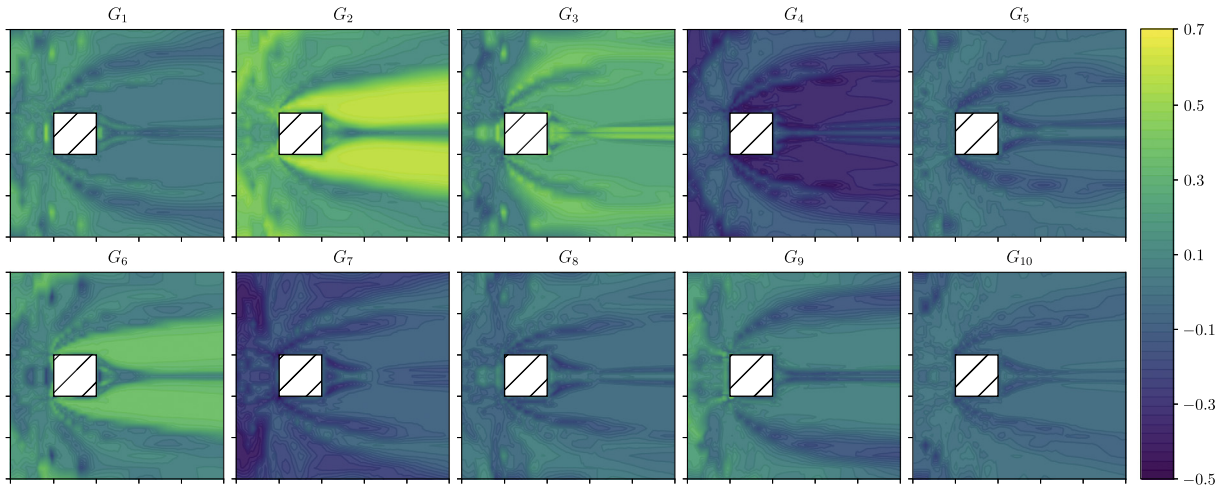

Fig. 5. The learned mixing coefficients of the training neural network for flow around a square cylinder [52] at bulk Reynolds number 5000. No domain reflections were used to artificially impose symmetry.

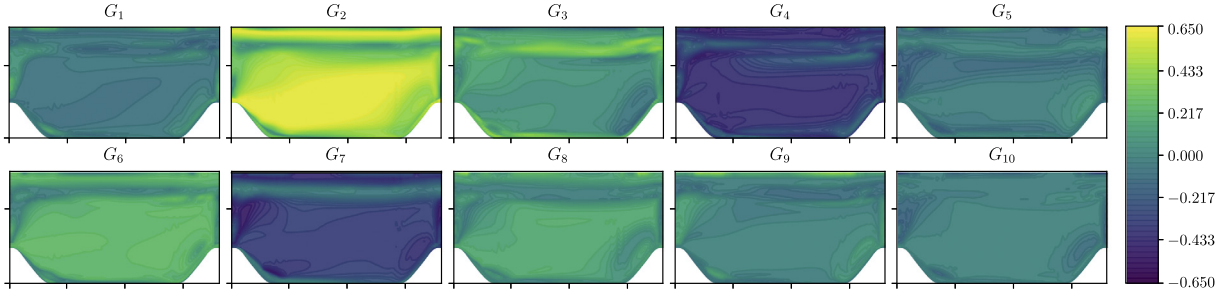

Fig. 6. The learned mixing coefficients of the training neural network for flow over periodic hills [53,54] at bulk Reynolds number 6210.

the treatment of domain boundaries as well as areas in which wall functions may be used. We address these issues using two different methods. First, the use of the baseline RANS TKE as a scaling factor of the anisotropic term shown in Eq. (11) allows for many turbulent boundary conditions to be satisfied. Second, to address areas in which wall functions may be used, we take inspiration from hybrid LES/RANS methods and introduce a blending function proposed by Xiao et al. [62]:

$$\boldsymbol{b}^* = \Gamma \boldsymbol{b}_{dd} + (1-\Gamma)\boldsymbol{b}_{rans}, \quad \Gamma = \tanh\left(d/\alpha_1 \lambda\right)^2, \tag{42}$$

where $\boldsymbol{b}_{dd}$ is the data-driven prediction of the anisotropic tensor, $\boldsymbol{b}_{rans}$ is the baseline RANS anisotropic tensor, $\lambda^2 = k/\epsilon$, d is the distance from the wall and α_1 is a tunable parameter. This function allows a smooth transition between the use of the baseline R-S near the wall and the data-driven prediction in the bulk flow. In the original work, it is suggested that the selection of α_1 be a value that achieves $\Gamma = 0.5$ somewhere is the log region. We found the value of 0.05 worked well for our test cases.

5. Numerical results

The use of data-driven models for test simulations whose domain is similar or identical to the training data is a frequent occurrence in the literature but does not correctly assess a data-driven model's performance. Since our selected neural network has already been shown to work adequately for similar flows in [17], our test cases are selected to deviate significantly from the training flows in both flow geometry and Reynolds number. We have selected the two test flows detailed in Table 2: flow over a backwards step and flow around a wall mounted cube both at three different Reynolds numbers. The geometry for each flow can be seen in Fig. 7. For both test cases, we will study the accuracy and respective uncertainty of both the model's R-S predictions as well as the predicted fluid quantities of interest. Ultimately, we wish to assess the predictive performance of the data-driven model for these geometrically different flows and use the proposed stochastic data-driven RANS algorithm to calculate probabilistic bounds on flow state variables by conducting uncertainty quantification on the data-driven turbulence model.

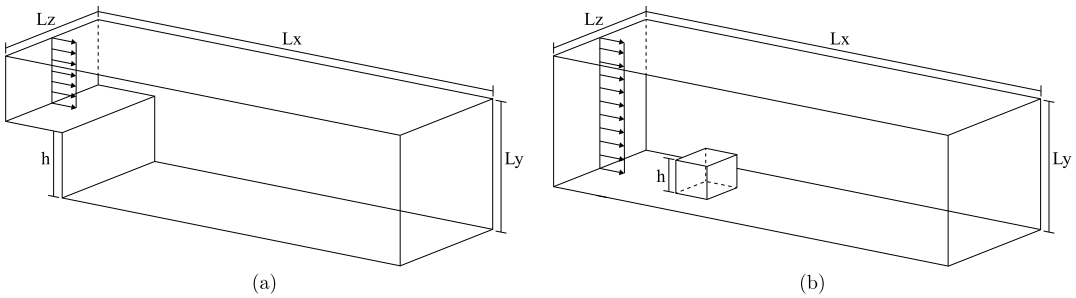

Fig. 7. (a) Flow geometry for the backwards step test flow with height h. (b) Flow geometry for the wall mounted cube test flow with height h.

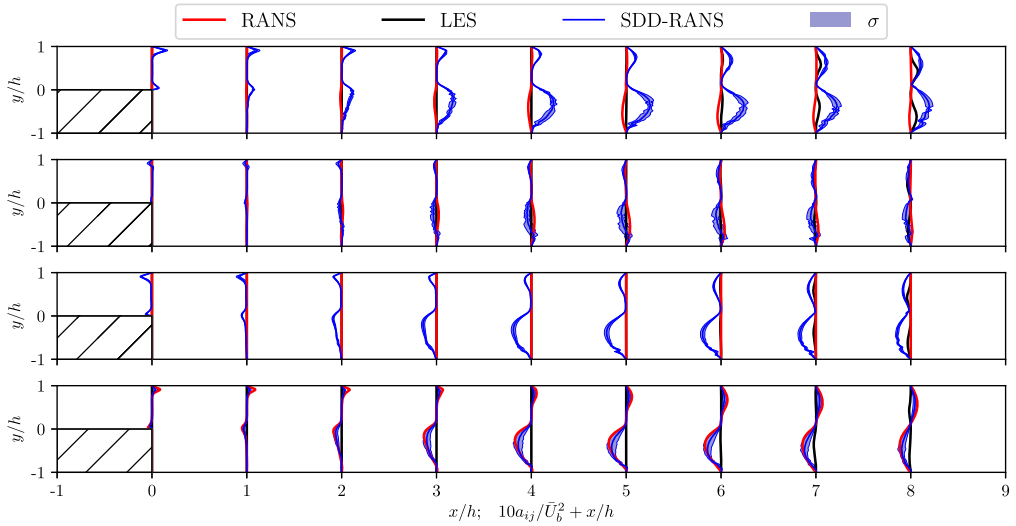

Fig. 8. The anisotropic term predictions for the backwards step test flow for Reynolds number 500. Top to bottom: a_{11}, a_{22}, a_{33} and a_{12} (a_{13} and a_{23} are omitted due to all fields being zero).

5.1. Backwards step

In the first test case, we select a backwards facing step at three different Reynolds numbers. As illustrated in Fig. 7a, this flow features a constant velocity inlet channel followed by a backwards facing step of height h. For this flow, we select the inlet channel to be the same height as the step. In contrast to most backwards step simulations, no-slip walls are on both the top and bottom faces. The z direction is periodic. On the x–y plane, we place the origin at the corner of the step. The flow features of interest are the recirculating regions that appear not only directly after the step but also on the upper channel wall down stream which is seen in the LES simulations in Figs. 11–13.

The predicted components of the unnormalized anisotropic term $\boldsymbol{a}$, defined by $\boldsymbol{a} = k \cdot \boldsymbol{b}$ where k is the baseline RANS TKE and $\boldsymbol{b}$ is the predicted anisotropic tensor, are shown in Figs. 8–10 for Reynolds numbers 500, 2500 and 5000. The first trend to notice is the relative consistency of SDD-RANS between all Reynolds numbers in terms of the magnitude of the mean predictions as well as variance. This is clearly an effect of the use of training data that vary little in Reynolds number compared to the test flows. For both a_{11} and a_{33} at Reynolds number 500 and 2500, the neural network is able to successfully predict the correct shape of the anisotropic term. This is a notable improvement of the baseline RANS prediction which severally under-predicts all normal components. For Reynolds number 5000, SDD-RANS favors only a single region for a_{11} and a_{33} as opposed to the two regions seen in lower Reynolds numbers. While these predictions are significant improvements over the baseline RANS solution, there are still key discrepancies including that the anisotropic components are consistently predicted upstream compared to the LES solution. Additionally, for terms such as a_{11}, the neural network under-predicts the magnitude for the larger Reynolds numbers as well as consistently under-predicts a_{22}. Briefly focusing on the epistemic uncertainty of the model's predictions, the variance of the neural network's predictions are relatively small indicating the model is over-confident for this test flow. Also we note that near the inlet (laminar region) there is little variance in the predicted anisotropic term. This shows that the neural network is able to identify regions that are more uncertain than others instead of just placing a uniform variance across the entire field.

The stream-wise velocity contours of the flow for the LES, baseline RANS and the expected velocity prediction of stochastic data-driven RANS (SDD-RANS) are depicted in Figs. 11–13. To keep plot labels uncluttered, we refer to the expected

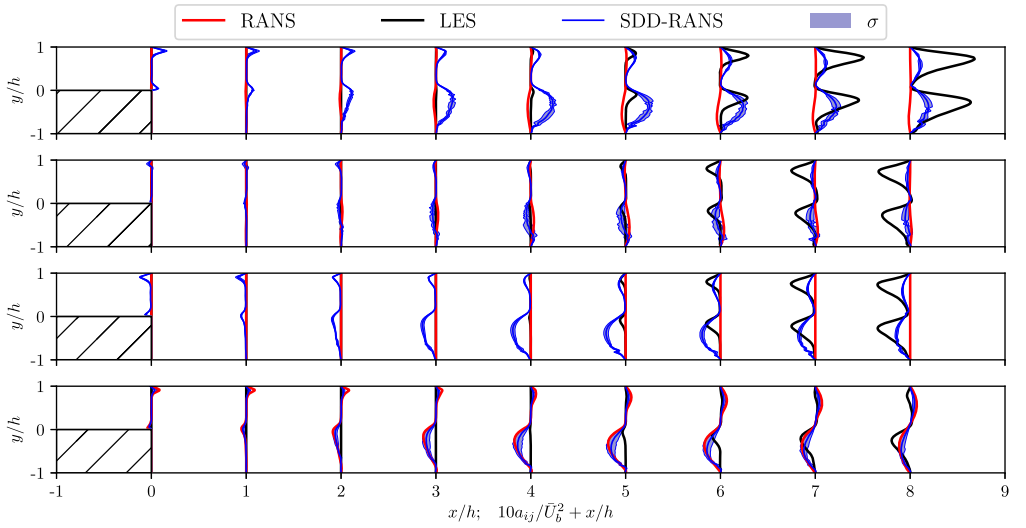

Fig. 9. The anisotropic term predictions for the backwards step test flow for Reynolds number 2500. Top to bottom: a_{11}, a_{22}, a_{33} and a_{12} (a_{13} and a_{23} are omitted due to all fields being zero).

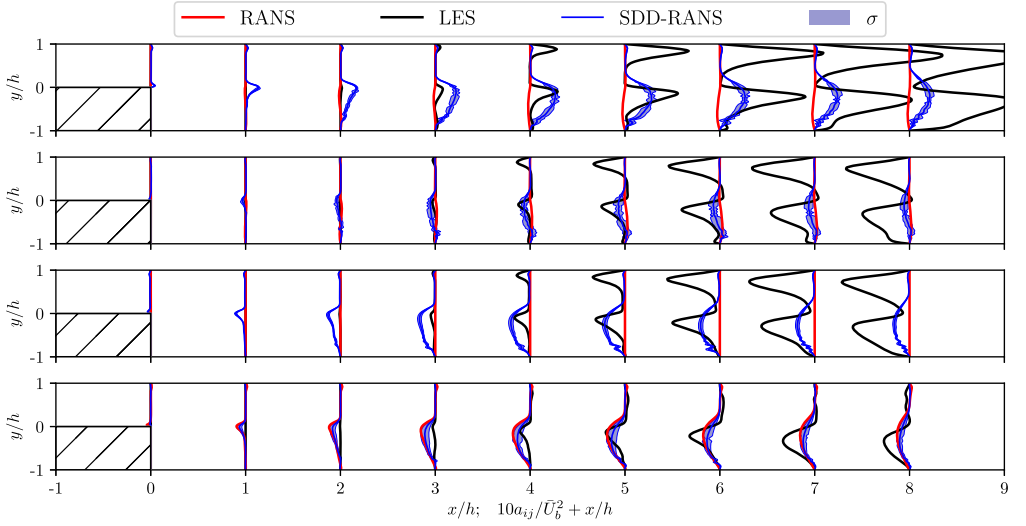

Fig. 10. The anisotropic term predictions for the backwards step test flow for Reynolds number 5000. Top to bottom: a_{11}, a_{22}, a_{33} and a_{12} (a_{13} and a_{23} are omitted due to all fields being zero).

values from the proposed framework as just SDD-RANS. For each Reynolds number, the mean stream-wise velocity profiles are also illustrated with the respective predictive error bars. We look first at the lowest Reynolds number of 500 for which the model produced the best prediction. Even though this corresponds to a Reynolds number furthest from the training data in Table 1, the stochastic model was able to successfully predict the appearance of the second recirculation region. The baseline RANS simulation only predicted a single eddy behind the step. While the anisotropic predictions are far larger in magnitude compared to LES, the viscous forces are large enough to correct these discrepancies at lower Reynolds numbers. We presume that these upstream over-predictions of the anisotropic terms in magnitude are the reason the second eddy appears closer to the inlet for SDD-RANS compared to the LES solution. Overall, for this lower Reynolds number, the model has little variance in its predictions since the effects of the R-S prediction are dampened.

As the Reynolds number increases, the role of the R-S increases significantly as the viscous forces weaken and a very clear degradation in the predictive performance of SDD-RANS is seen. For Reynolds number 2500 and 5000 (see Figs. 12–13), SDD-RANS does not yield any prediction improvement over the baseline RANS simulation with the exception of the recirculating region near the step ($x/h \leq 7.5$) where SDD-RANS is able to accurately predict the flow.

For both higher Reynolds numbers test cases, SDD-RANS fails to predict the upper recirculation region that is present in the LES solution. This is likely due to the under-prediction of the anisotropic components downstream seen in Figs. 9–10. As the R-S is increased, minor deviations in the anisotropic term are amplified resulting in potentially starkly different

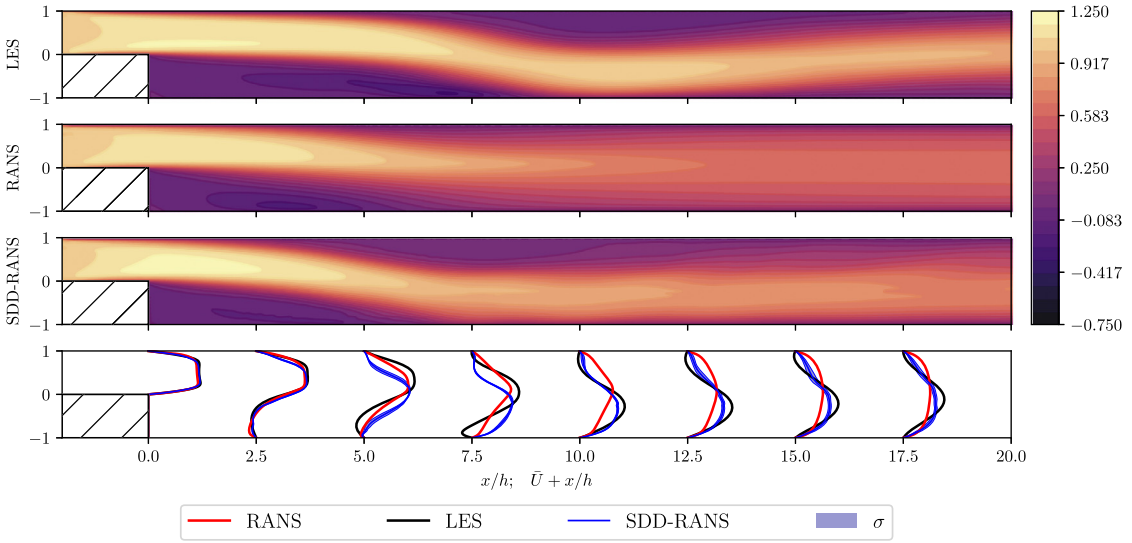

Fig. 11. Normalized stream-wise mean velocity contours for Reynolds number 500. The top is the LES solution, below is the baseline RANS prediction followed by the data-driven mean field. Lastly is the stream-wise mean velocity profiles for all simulations shown along with the predictive error bars of the SDD-RANS prediction.

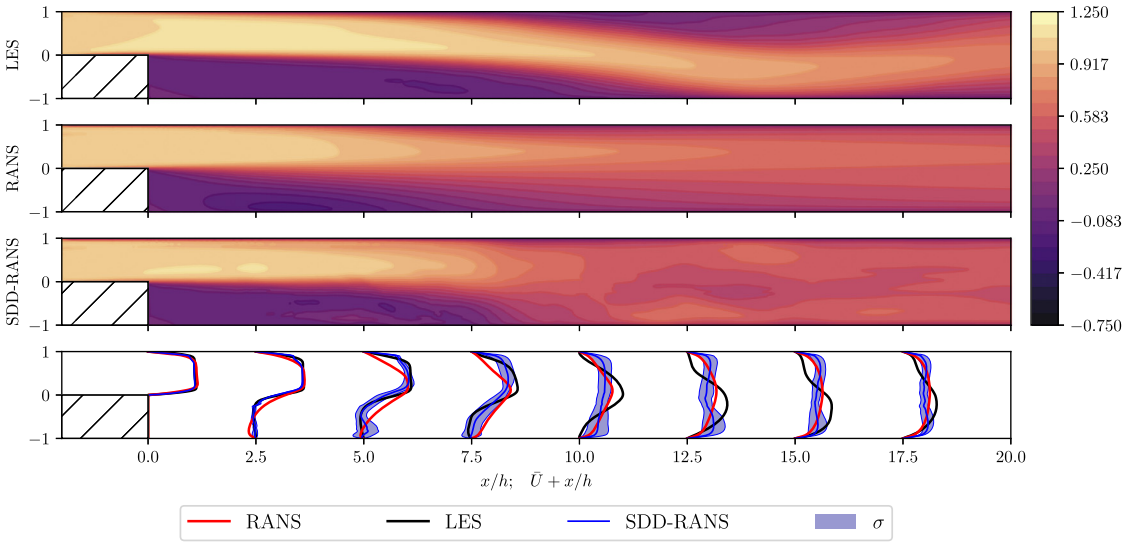

Fig. 12. Normalized stream-wise mean velocity contours for Reynolds number 2500. The top is the LES solution, below is the baseline RANS prediction followed by the data-driven mean field. Lastly is the stream-wise mean velocity profiles for all simulations shown along with the predictive error bars of the SDD-RANS prediction.

flow predictions [27]. SDD-RANS accurately captures these phenemona. As the Reynolds number increases, so does the standard deviation indicating a loss of model confidence. In addition, it is clear that the model is extremely confident in its predictions towards the inlet where it is able to match the LES solution. However, this confidence quickly diminishes downstream as flow predictions become increasingly less accurate. With a deterministic data-driven model such indicators would not be present allowing for no interpretable information on prediction confidence without observed high-fidelity data.

5.2. Wall mounted cube

The second test case is flow around a wall mounted cube with height h as shown in Fig. 7b. Unlike the majority of the flows that have been tested by data-driven models in the literature [13,15–17,20] as well as our training flows, this test flow contains an obstacle that is not semi-infinite. This means that flow with this geometry cannot be modeled by a two-dimensional RANS simulation as was the case for all previously considered flows. Additionally, similar to the backwards

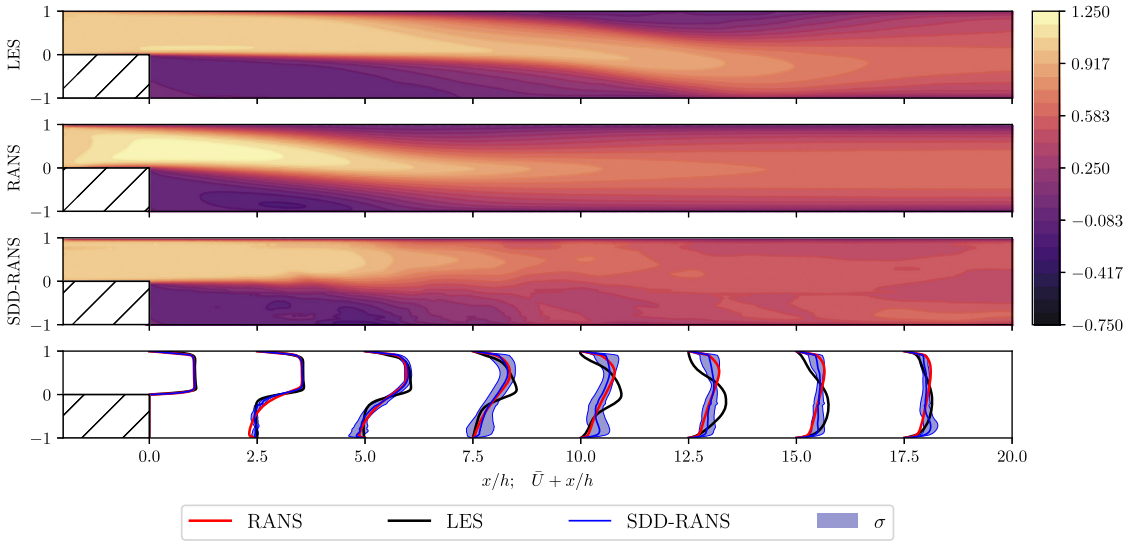

Fig. 13. Normalized stream-wise mean velocity contours for Reynolds number 5000. The top is the LES solution, below is the baseline RANS prediction followed by the data-driven mean field. Lastly is the stream-wise mean velocity profiles for all simulations shown along with the predictive error bars of the SDD-RANS prediction.

step, none of our training flows contain a geometry that is similar to this. As a result, we consider this flow an excellent test to investigate the limits of SDD-RANS in generalizing to a true 3D test case.

The set-up of this flow consists of an uniform inlet velocity and two channel walls normal to the y-axis. The cube is placed slightly down stream of the inlet. The feature of interest is primarily the recirculation region behind the cube itself. Additionally, as the Reynolds number increases, flow separation occurs on the sides of the cube. As will be shown in the subsequent figures, this flow separation is often non-existent for the baseline RANS predictions. While the mean flow is symmetrical about the x–y plane in the middle of the channel ($z = 3.5H$), we simulate the entire cube in order to observe non-symmetrical behavior in predictions.

The prediction of the unnormalized anisotropic term along the x–y plane of symmetry is shown in Fig. 14 for Reynolds number 2500. For brevity, we only show the results for a single Reynolds number since both Reynolds numbers 500 and 5000 lead to similar predictions. From this figure, several positive traits are seen for both the mean neural network predictions as well as the associated uncertainty. The bulk region shows little variability within the predictive error bounds. Instead the variance is largely concentrated behind the obstacle. This is a nice attribute because the baseline RANS simulation is accurate in the bulk region, thus perturbing the R-S in the bulk flow would not be of any benefit. Another interesting feature predicted by the neural network is the concentrated region of normal stresses on top of the cube seen in a_{11} and a_{33}. Similar regions, yet smaller in magnitude, appear further down-stream in the LES solution as a result of the shear layer that forms between the bulk and recirculation regions. In addition, the SDD-RANS is able to significantly improve the R-S prediction towards the front of the cube where the baseline RANS solution is incorrect. This includes the leading corner of the cube at $x = 3$ where SDD-RANS is able to largely correct the baseline RANS solution which has sharp, unphysical predictions near the edge.

Despite the improvements to the upstream edge of the cube, the expected value of SDD-RANS has no marginal improvement on the baseline RANS prediction in the recirculation region. However, as reflected in the predictive error bounds, the variance is larger in the recirculation zone often being able to enclose part of the true LES solution. This is a promising result because, although the mean predictions have not improved, the model is uncertain regarding its predictions in this region. This suggests that this area of the flow could contain physics the model has not seen before in the training data. The stream-wise velocity contours on the plane of symmetry for the LES, baseline RANS and the expected value of SDD-RANS are depicted below in Fig. 15. Similar to the backwards step, as the Reynolds number increases, the standard deviation of the SDD-RANS prediction also increases. However, the magnitude of the variance is significantly smaller than that of the backwards step for higher Reynolds numbers reflecting the more accurate predictions for this problem.

To take a closer look at the performance of SDD-RANS, stream-wise velocity profiles are plotted for both Reynolds number 500 and 5000 in Figs. 16 and 17, respectively. For each, a plot containing the resulting SDD-RANS velocity field samples are shown as well as the predictive standard deviation error bars. As expected the variance in the velocity samples increases with the increased Reynolds number, however this only occurs in the recirculation region where the instantaneous flow is turbulent. In the bulk region above the recirculation zone, the variance remains small.

A significant improvement by SDD-RANS for both Reynolds numbers is the prediction of the detached flow on top of the cube ($x/h = 4$) which the baseline RANS fails to capture. This is largely due to the large increase of normal stresses from the neural network predictions around the surrounding walls of the cube which results in the shear layer forming above

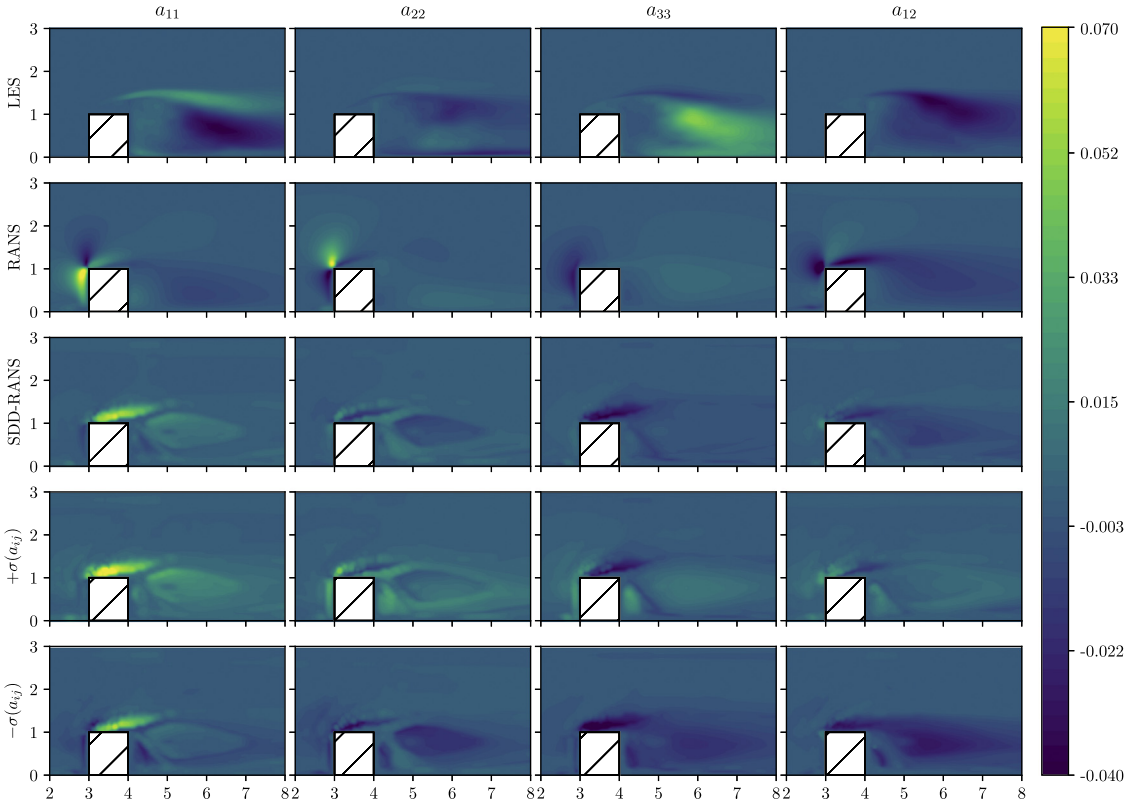

Fig. 14. The anisotropic term predictions for the wall mounted cube test flow for Reynolds number 2500 along the plane of symmetry (a_{13} and a_{23} are omitted because both fields being zero on the plane of symmetry). From top to bottom: Time-averaged LES solution, baseline RANS prediction, SDD-RANS expected value and the predictive standard deviation error bounds.

the obstacle. For both Reynolds numbers cases, improvements in the recirculation region prediction are present. However, for the lower Reynolds number case SDD-RANS leads to more accurate predictions.

Since this obstacle is not semi-infinite, the velocity contours for the horizontal plane at $y = 0.5h$ are shown in Fig. 18. Similarly, velocity profiles are plotted for Reynolds number 2500 in Fig. 19. In general, we can see the same trends as previous results for which the variance increases with Reynolds number. We note that SDD-RANS is able to predict the presence of the detached flow on the side of the cube. While some asymmetry exists in the SDD-RANS predictions, the predictions overall retain a general symmetric profile. It is likely that increasing the number of samples of R-S fields would further improve the symmetry and smoothness of predictions.

This framework allows probabilistic bounds to be calculated for other fluid properties such as pressure, drag, shear stress, etc. For example, two pressure profiles along the face of the wall mounted cube are plotted in Figs. 20 and 21. In general we see that SDD-RANS is able to provide an improved prediction compared to the baseline RANS. Similar to the anisotropic components, SDD-RANS corrects the unphysical pressure drop that occurs on the edge of the leading cube face in the baseline RANS simulation. The uncertainty for the predictive pressure is also very reasonable nearly capturing the true LES prediction for all faces. Similar to the velocity predictions in Figs. 15 and 18, the variance of the pressure on the upstream face of the cube is significantly smaller reflecting the model's confidence in this laminar region.

6. Conclusions

As the CFD community continues to investigate the use of machine learning tools for data-driven modeling, the need to accurately quantify the induced uncertainties from the use of such models becomes essential. In this work, we have presented a novel framework that allows for the quantification of such model form uncertainty. To satisfy invariant properties, we use the neural network architecture originally proposed by Ling et al. [17]. Using Stein variational gradient decent and following the work of Zhu and Zabaras [31], we extended this invariant neural network model to a Bayesian deep neural network to allow us to compute the distribution of the anisotropic R-S stress tensor for a given baseline solution. To propagate the uncertainty of this model to fluid flow quantities of interest, a stochastic data-driven RANS algorithm is proposed that utilizes standard Monte Carlo simulation. The integrated framework was rigorously investigated on two flows to observe its generalization property.

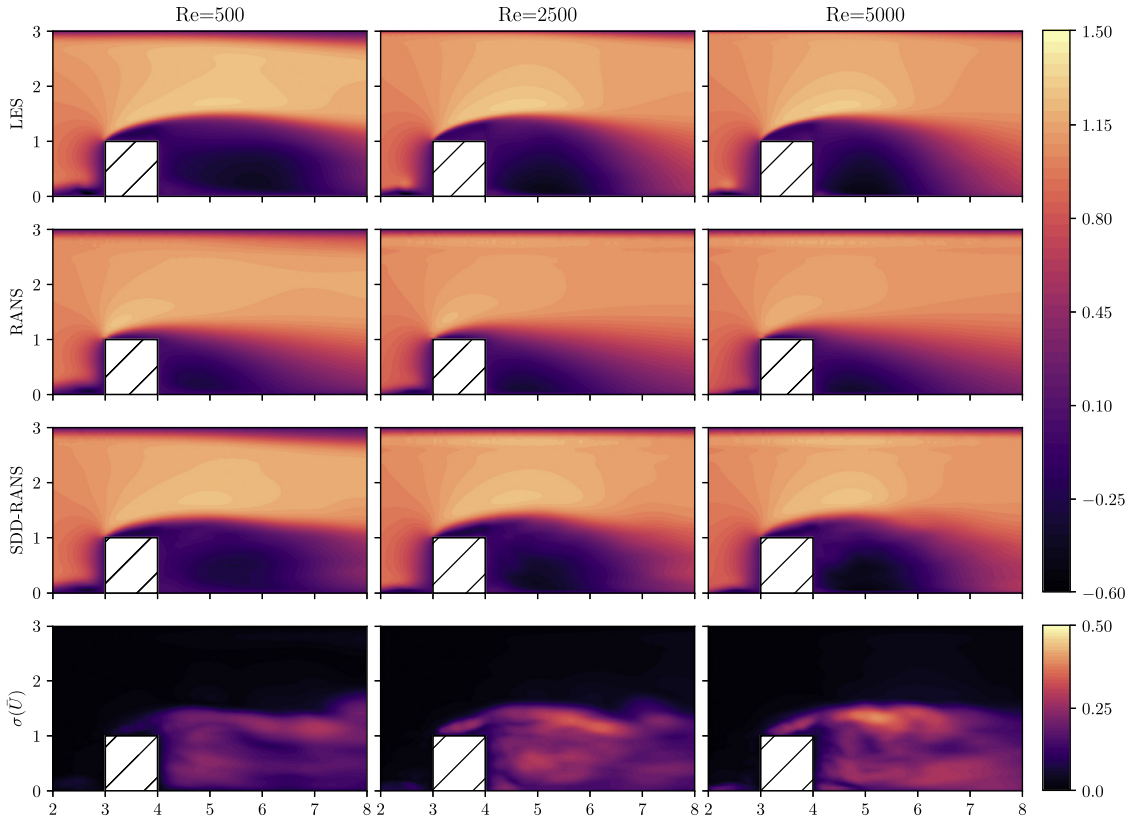

Fig. 15. Normalized stream-wise mean velocity contours for Reynolds numbers 500, 2500 and 5000 on the plane of symmetry. The top is the time averaged LES solution, below is the baseline RANS prediction followed by the SDD-RANS expected velocity. The fourth row shows the standard deviation field of the data-driven prediction.

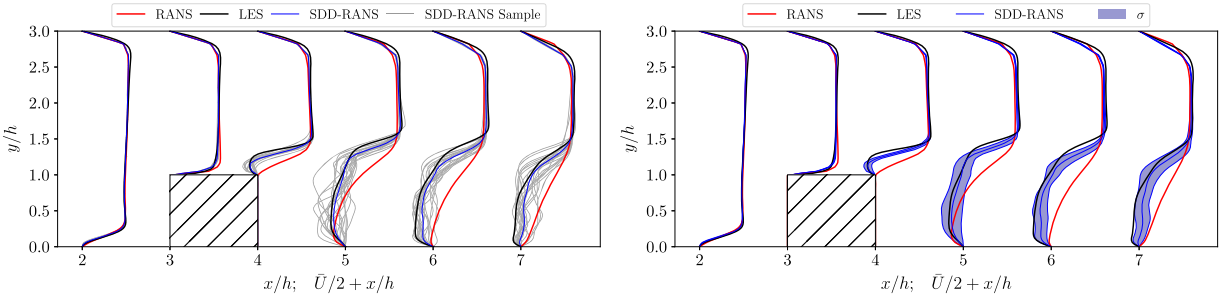

Fig. 16. Normalized stream-wise velocity profiles for the baseline RANS, high-fidelity LES, and SDD-RANS predictions on the plane of symmetry at six different locations in the stream-wise direction for $Re = 500$. The left shows the SDD-RANS velocity samples, and the right shows the respective predictive error bars for each profile.

In the presented implementation of this framework, we found that the invariant neural network used to model the anisotropic tensor proved difficult to train and yield satisfactory predictions for unseen flows and geometries. From our studies, we hypothesize that using just the five invariant inputs does not provide enough descriptive information to accurately map from the coarse to high-fidelity flow physics. Although the network contains desired invariant properties, other flow quantities would most likely need to be used as model inputs to improve the quality of predictions. Thus a critical area to be investigated is the development of more accurate Reynolds stress representations by identifying the important local- and non-local variables that influence its values. The potential use of spatial correlations and information at neighboring nodes (non-local models) may prove to be extremely beneficial. Such an approach, while difficult to implement for non-uniform grids, can be easily applied in the context of convolutional neural networks that are capable of mapping many high-dimensional inputs to multi-outputs of high-dimensionality. While a number of other models in the literature can yield much better training predictions, most of these models remain only useful to a small family of flows resembling those in the training dataset. The generalization property of these models remains an open problem for the data-driven community.

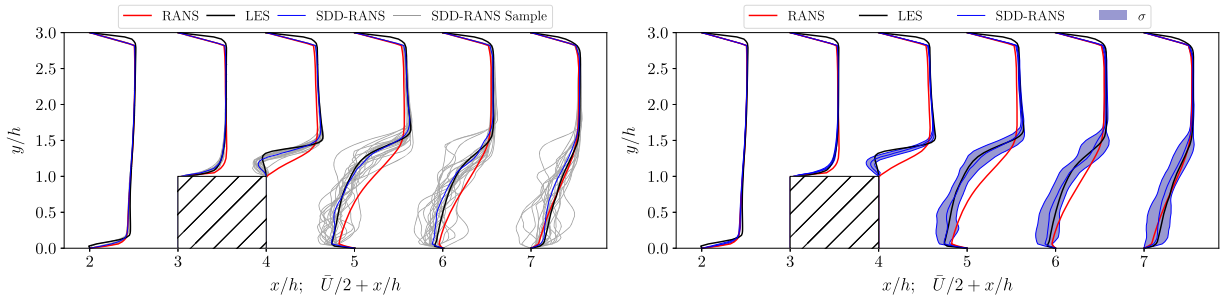

Fig. 17. Normalized stream-wise velocity profiles for the baseline RANS, high-fidelity LES, and SDD-RANS predictions on the plane of symmetry at six different locations in the stream-wise direction for $Re = 5000$. The left shows the SDD-RANS velocity samples, and the right shows the respective predictive error bars for each profile.

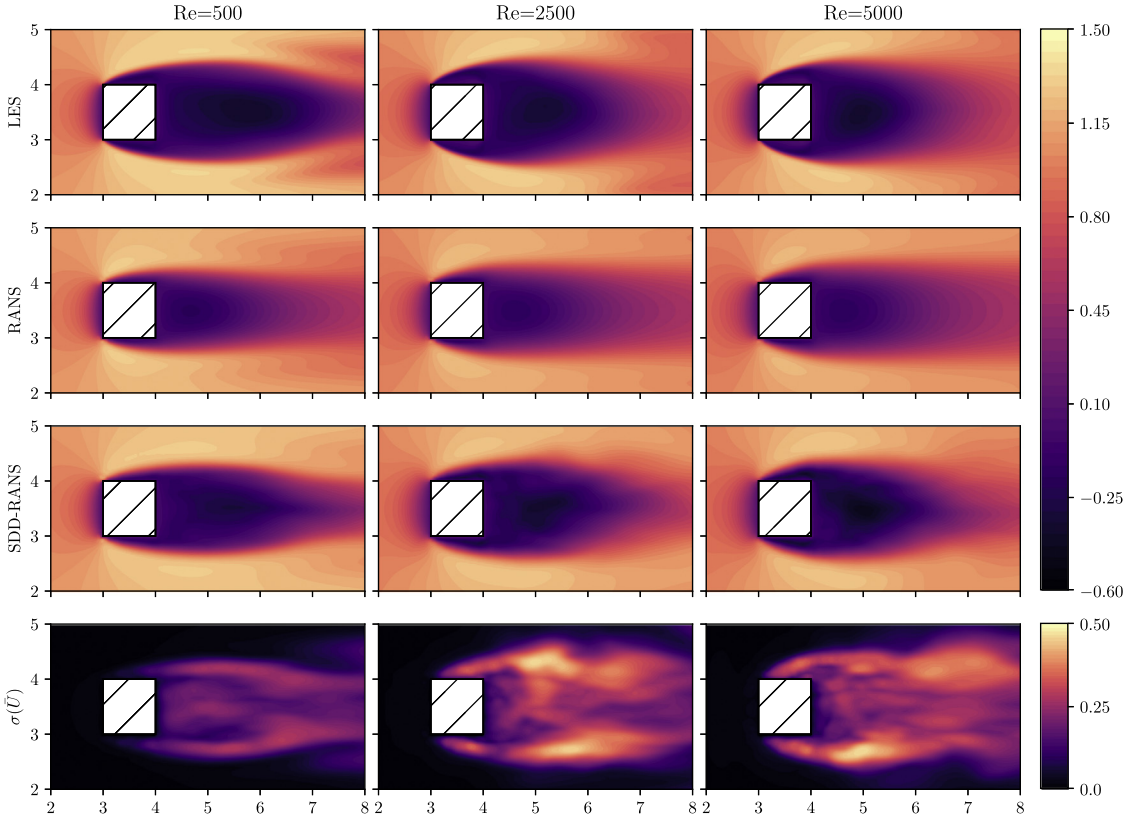

Fig. 18. Normalized stream-wise mean velocity contours for Reynolds numbers 500, 2500 and 5000 on the $y = 0.5h$ plane. The top is the time averaged LES solution, below is the baseline RANS prediction followed by the SDD-RANS expected velocity. The fourth row shows the standard deviation field of the data-driven prediction.

With improvements in the representation of the tuned Reynolds stress in the RANS equations, we believe that this framework can provide extremely beneficial information for data-driven models. While the most obvious application is its use to assess a given model's predictive confidence, the developed framework can also be used to identify locations of potentially lower accuracy. This could be useful for identifying areas that may require finer mesh resolutions or high-fidelity simulations. Additionally, the use of a Bayesian neural network allows us to compute predictive bounds for the quantities of interest. This can be extremely useful in cases where the training data is limited. Even though the use of the Bayesian neural network and SDD-RANS requires more computational time than deterministic data-driven approaches, we found that when compared to high-fidelity simulations the computational cost remains low.

The use of a Bayesian neural network opens up the potential of implementing experimental design techniques by investigating the impact of training data on the quality of the model's predictions. This can range from the assessment of a limited data case or how specific training flows at various Reynolds numbers impact a specific test case prediction. Finally, a detailed analysis of epistemic uncertainty would be beneficial to the data-driven turbulence modeling community.

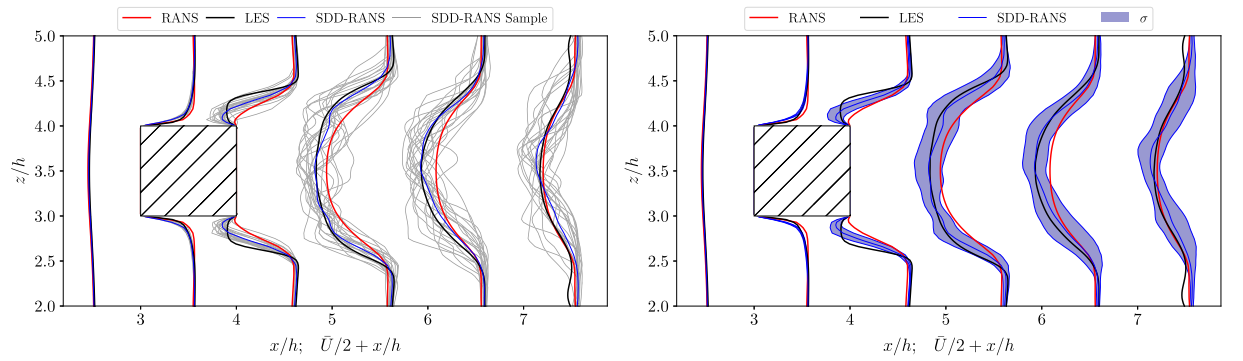

Fig. 19. Normalized stream-wise velocity on the $y = 0.5h$ plane for the baseline RANS, high-fidelity LES, and SDD-RANS predictions on the plane of symmetry at six different locations in the stream-wise direction for $Re = 2500$. The left shows the SDD-RANS velocity samples, and the right shows the respective predictive error bars for each profile.

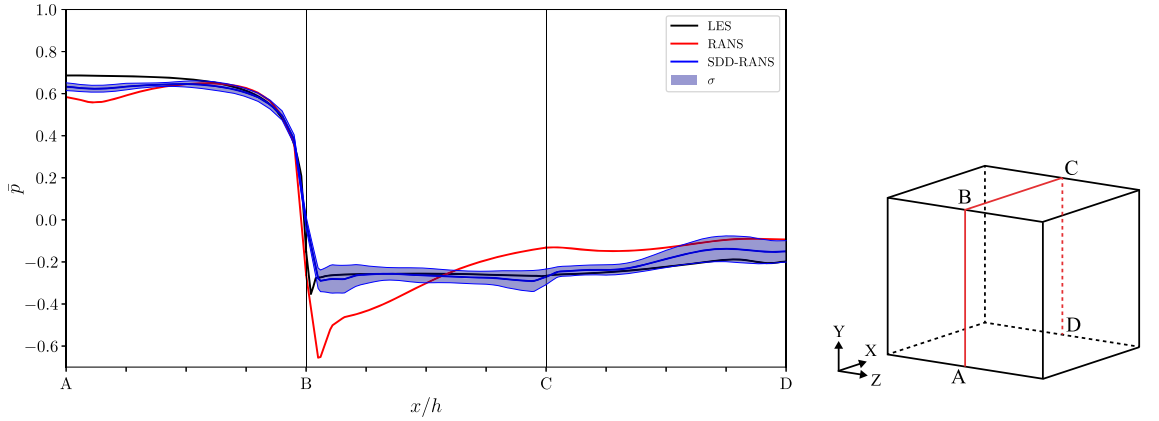

Fig. 20. Normalized mean surface pressure profile on the plane of symmetry ($z = 3.5h$) for $Re = 5000$.

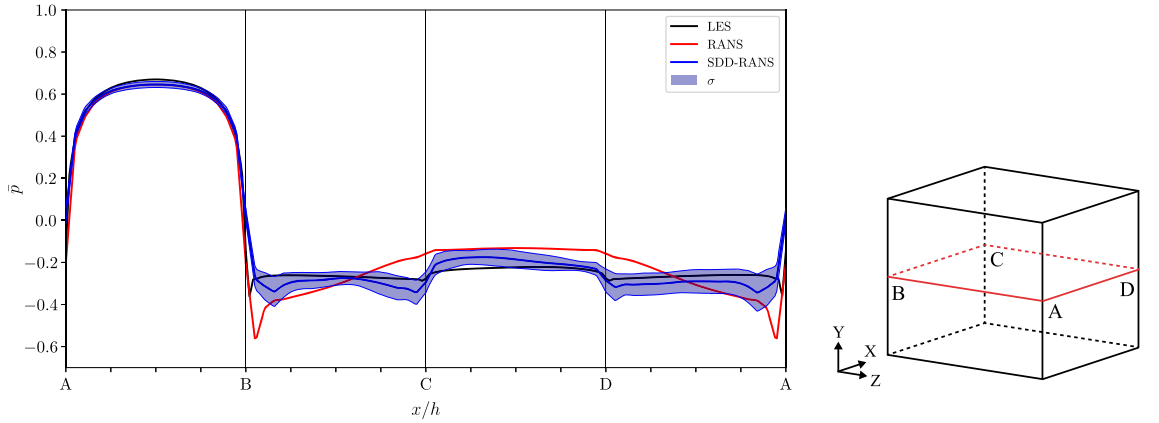

Fig. 21. Normalized mean surface pressure profile on the plane $y = 0.5h$ for $Re = 5000$.

Acknowledgements

The authors acknowledge support from the Defense Advanced Research Projects Agency (DARPA) under the Physics of Artificial Intelligence (PAI) program (grant No. HR00111890034). The work of NG is also supported by a National Science Foundation (NSF) Graduate Research Fellowship Program grant No. DGE-1313583. The computing was facilitated by the resources of the NSF supported "Extreme Science and Engineering Discovery Environment" (XSEDE) on the Bridges and Bridges-GPU cluster through the startup allocation No. TG-CTS180011 and research allocation No. TG-CTS180038. Additional computing resources were provided by the University of Notre Dame's Center for Research Computing (CRC).

References

[1] S.B. Pope, Turbulent Flows, Cambridge University Press, Cambridge, 2000.
[2] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, in: 30th Aerospace Sciences Meeting and Exhibit, 1992, p. 439.
[3] P. Godin, D. Zingg, T. Nelson, High-lift aerodynamic computations with one-and two-equation turbulence models, AIAA J. 35 (2) (1997) 237–243, https://doi.org/10.2514/2.113.
[4] W. Jones, B. Launder, The prediction of laminarization with a two-equation model of turbulence, Int. J. Heat Mass Transf. 15 (2) (1972) 301–314, https://doi.org/10.1016/0017-9310(72)90076-2.
[5] B. Launder, B. Sharma, Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc, Lett. Heat Mass Transf. 1 (2) (1974) 131–137, https://doi.org/10.1016/0094-4548(74)90150-7.
[6] D.C. Wilcox, Turbulence Modelling for CFD, DCW Industries, La Cañada, 1993.
[7] F.R. Menter, Two-equation eddy-viscosity turbulence models for engineering applications, AIAA J. 32 (8) (1994) 1598–1605, https://doi.org/10.2514/3.12149.
[8] D.K. Walters, D. Cokljat, A three-equation eddy-viscosity model for Reynolds-averaged Navier–Stokes simulations of transitional flow, J. Fluids Eng. 130 (12) (2008) 121401, https://doi.org/10.1115/1.2979230.
[9] M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. 182 (1) (2002) 1–26, https://doi.org/10.1006/jcph.2002.7146.
[10] S.H. Cheung, T.A. Oliver, E.E. Prudencio, S. Prudhomme, R.D. Moser, Bayesian uncertainty analysis with applications to turbulence modeling, Reliab. Eng. Syst. Saf. 96 (9) (2011) 1137–1149, https://doi.org/10.1016/j.ress.2010.09.013.
[11] T.A. Oliver, R.D. Moser, Bayesian uncertainty quantification applied to RANS turbulence models, J. Phys. Conf. Ser. 318 (2011) 042032, IOP Publishing, http://stacks.iop.org/1742-6596/318/i=4/a=042032.
[12] E. Dow, Q. Wang, Uncertainty quantification of structural uncertainties in RANS simulations of complex flows, in: 20th AIAA Computational Fluid Dynamics Conference, 2011, p. 3865.
[13] B. Tracey, K. Duraisamy, J. Alonso, Application of supervised learning to quantify uncertainties in turbulence and combustion modeling, in: 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2013, p. 259.
[14] B.D. Tracey, K. Duraisamy, J.J. Alonso, A machine learning strategy to assist turbulence model development, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1287.
[15] A.P. Singh, S. Medida, K. Duraisamy, Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils, AIAA J. (2017) 1–13, https://doi.org/10.2514/1.J055595.
[16] Z.J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2460.
[17] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166, https://doi.org/10.1017/jfm.2016.615.
[18] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, J. Comput. Phys. 318 (2016) 22–35, https://doi.org/10.1016/j.jcp.2016.05.003.
[19] J. Ling, A. Ruiz, G. Lacaze, J. Oefelein, Uncertainty analysis and data-driven model advances for a jet-in-crossflow, J. Turbomach. 139 (2) (2017) 021008, https://doi.org/10.1115/1.4034556.
[20] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: a data-driven, physics-informed Bayesian approach, J. Comput. Phys. 324 (2016) 115–136, https://doi.org/10.1016/j.jcp.2016.07.038.
[21] J.-L. Wu, J.-X. Wang, H. Xiao, J. Ling, A priori assessment of prediction confidence for data-driven turbulence modeling, Flow Turbul. Combust. 99 (1) (2017) 25–46, https://doi.org/10.1007/s10494-017-9807-0.
[22] I. Boussinesq, Essai sur la théorie des eaux courantes, Mem. Pres. par div. savants a l'Acad. Sci. Paris, vol. 23, 1877, p. 46.
[23] B.E. Launder, D.B. Spalding, The numerical computation of turbulent flows, in: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, Elsevier, 1983, pp. 96–116.
[24] K.-Y. Chien, Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model, AIAA J. 20 (1) (1982) 33–38, https://arc.aiaa.org/doi/abs/10.2514/3.51043.
[25] F. Menter, Zonal two equation $k-\omega$ turbulence models for aerodynamic flows, in: 23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference, 1993, p. 2906.
[26] R.L. Thompson, L.E.B. Sampaio, F.A. de Bragança Alves, L. Thais, G. Mompean, A methodology to evaluate statistical errors in DNS data of plane channel flows, Comput. Fluids 130 (2016) 1–7, https://doi.org/10.1016/j.compfluid.2016.01.014.
[27] J. Wu, H. Xiao, R. Sun, Q. Wang, RANS equations with Reynolds stress closure can be ill-conditioned, arXiv preprint, arXiv:1803.05581.
[28] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436, https://doi.org/10.1038/nature14539.
[29] S. Pope, A more general effective-viscosity hypothesis, J. Fluid Mech. 72 (2) (1975) 331–340, https://doi.org/10.1017/S0022112075003382.
[30] C.M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2006.
[31] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447, https://doi.org/10.1016/j.jcp.2018.04.018.
[32] D.J. MacKay, Bayesian Methods for Adaptive Models, Ph.D. thesis, California Institute of Technology, 1992.
[33] R.M. Neal, Bayesian Learning for Neural Networks, vol. 118, Springer Science & Business Media, 2012, https://www.springer.com/us/book/9780387947242.
[34] M.D. Richard, R.P. Lippmann, Neural network classifiers estimate Bayesian a posteriori probabilities, Neural Comput. 3 (4) (1991) 461–483, https://doi.org/10.1162/neco.1991.3.4.461.
[35] D. Barber, C.M. Bishop, Ensemble Learning in Bayesian Neural Networks, NATO ASI Series of Computer and Systems Sciences, vol. 168, 1998, pp. 215–238, https://www.microsoft.com/en-us/research/publication/ensemble-learning-in-bayesian-neural-networks/.
[36] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, arXiv preprint, arXiv:1505.05424.
[37] D.P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, in: Advances in Neural Information Processing Systems, 2015, pp. 2575–2583, https://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.
[38] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning – vol. 48, ICML'16, 2016, pp. 1050–1059, http://dl.acm.org/citation.cfm?id=3045390.3045502.
[39] Q. Liu, D. Wang, Stein variational gradient descent: a general purpose Bayesian inference algorithm, in: Advances In Neural Information Processing Systems, 2016, pp. 2378–2386.
[40] Q. Liu, Stein variational gradient descent as gradient flow, in: Advances in Neural Information Processing Systems, 2017, pp. 3118–3126, https://papers.nips.cc/paper/6338-stein-variational-gradient-descent-a-general-purpose-bayesian-inference-algorithm.pdf.
[41] M.E. Tipping, Sparse Bayesian learning and the relevance vector machine, J. Mach. Learn. Res. 1 (Jun. 2001) 211–244, http://www.jmlr.org/papers/v1/tipping01a.html.

[42] J.L. Beck, L.S. Katafygiotis, Updating models and their uncertainties. I: Bayesian statistical framework, J. Eng. Mech. 124 (4) (1998) 455–461, https://doi.org/10.1061/(ASCE)0733-9399(1998)124:4(455).
[43] J.L. Beck, S.-K. Au, Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation, J. Eng. Mech. 128 (4) (2002) 380–391, https://doi.org/10.1061/(ASCE)0733-9399(2002)128:4(380).
[44] S.H. Cheung, J.L. Beck, Bayesian model updating using hybrid Monte Carlo simulation with application to structural dynamic models with many uncertain parameters, J. Eng. Mech. 135 (4) (2009) 243–255, https://doi.org/10.1061/(ASCE)0733-9399(2009)135:4(243).
[45] H.G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Comput. Phys. 12 (6) (1998) 620–631, https://doi.org/10.1063/1.168744.
[46] H. Jasak, A. Jemcov, Z. Tukovic, et al., OpenFOAM: a C++ library for complex physics simulations, in: International Workshop on Coupled Methods in Numerical Dynamics, vol. 1000, IUC, Dubrovnik, Croatia, 2007, pp. 1–20.
[47] S.V. Patankar, D.B. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, in: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, Pergamon, 1983, pp. 54–73.
[48] R.I. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, J. Comput. Phys. 62 (1) (1986) 40–65, https://doi.org/10.1016/0021-9991(86)90099-9.
[49] J. Smagorinsky, General circulation experiments with the primitive equations: I. The basic experiment, Mon. Weather Rev. 91 (3) (1963) 99–164, https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.
[50] L.A. Schiavo, A.B. Jesus, J.L. Azevedo, W.R. Wolf, Large Eddy Simulations of convergent–divergent channel flows at moderate Reynolds numbers, Int. J. Heat Fluid Flow 56 (2015) 137–151, https://doi.org/10.1016/j.ijheatfluidflow.2015.07.006.
[51] Research Langley, Center turbulence modeling resource LES: 2-D converging–diverging channel, https://turbmodels.larc.nasa.gov/Other_LES_Data/conv-div-channel20580les.html. (Accessed 11 April 2018).
[52] G. Bosch, W. Rodi, Simulation of vortex shedding past a square cylinder with different turbulence models, Int. J. Numer. Methods Fluids 28 (4) (1998) 601–616, https://doi.org/10.1002/(SICI)1097-0363(19980930)28:4<601::AID-FLD732>3.0.CO;2-F.
[53] L. Temmerman, M.A. Leschziner, C.P. Mellen, J. Fröhlich, Investigation of wall-function approximations and subgrid-scale models in large eddy simulation of separated flow in a channel with streamwise periodic constrictions, Int. J. Heat Fluid Flow 24 (2) (2003) 157–180, https://doi.org/10.1016/S0142-727X(02)00222-9.
[54] Research Langley, Center turbulence modeling resource LES: 2-D periodic hill, https://turbmodels.larc.nasa.gov/Other_LES_Data/2dhill_periodic.html. (Accessed 11 April 2018).
[55] A. Pinelli, M. Uhlmann, A. Sekimoto, G. Kawahara, Reynolds number dependence of mean flow structure in square duct turbulence, J. Fluid Mech. 644 (2010) 107–122, https://doi.org/10.1017/S0022112009992242.
[56] H. Gopalan, R. Jaiman, Numerical study of the flow interference between tandem cylinders employing non-linear hybrid URANS–LES methods, J. Wind Eng. Ind. Aerodyn. 142 (2015) 111–129, https://doi.org/10.1016/j.jweia.2015.03.017.
[57] P.M. Gresho, D.K. Gartling, J. Torczynski, K. Cliffe, K. Winters, T. Garratt, A. Spence, J.W. Goodrich, Is the steady viscous incompressible two-dimensional flow over a backward-facing step at $Re = 800$ stable?, Int. J. Numer. Methods Fluids 17 (6) (1993) 501–541, https://doi.org/10.1002/fld.1650170605.
[58] A. Yakhot, H. Liu, N. Nikitin, Turbulent flow around a wall-mounted cube: a direct numerical simulation, Int. J. Heat Fluid Flow 27 (6) (2006) 994–1009, https://doi.org/10.1016/j.ijheatfluidflow.2006.02.026.
[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch.
[60] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980.
[61] H. Li, C.P. Chen, H.-P. Huang, Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering, CRC Press, 2000.
[62] X. Xiao, J. Edwards, H. Hassan, Blending functions in hybrid large-eddy/Reynolds-averaged Navier–Stokes simulations, AIAA J. 42 (12) (2004) 2508–2515, https://doi.org/10.2514/1.2094.