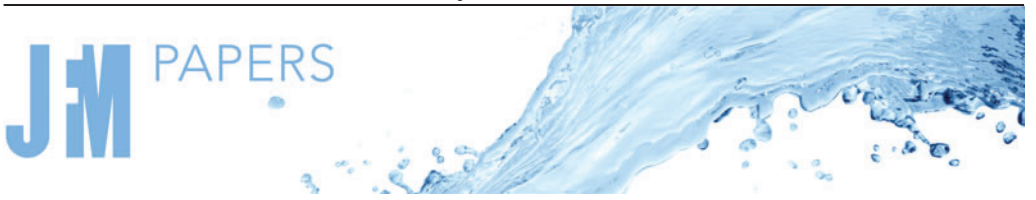
*J. Fluid Mech.* (2022), *vol*. 949, A26, doi:10.1017/jfm.2022.744



# Ensemble Kalman method for learning turbulence models from indirect observation data

**Xin-Lei Zhang**[1,2], **Heng Xiao**[3,†], **Xiaodong Luo**[4] **and Guowei He**[1,2,†]

[1]The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100049, PR China

[2]School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, PR China

[3]Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA

[4]Norwegian Research Centre (NORCE), Nygårdsgaten 112, 5008 Bergen, Norway

(Received 10 February 2022; revised 15 July 2022; accepted 21 August 2022)

In this work, we propose using an ensemble Kalman method to learn a nonlinear eddy viscosity model, represented as a tensor basis neural network, from velocity data. Data-driven turbulence models have emerged as a promising alternative to traditional models for providing closure mapping from the mean velocities to Reynolds stresses. Most data-driven models in this category need full-field Reynolds stress data for training, which not only places stringent demand on the data generation but also makes the trained model ill-conditioned and lacks robustness. This difficulty can be alleviated by incorporating the Reynolds-averaged Navier–Stokes (RANS) solver in the training process. However, this would necessitate developing adjoint solvers of the RANS model, which requires extra effort in code development and maintenance. Given this difficulty, we present an ensemble Kalman method with an adaptive step size to train a neural-network-based turbulence model by using indirect observation data. To our knowledge, this is the first such attempt in turbulence modelling. The ensemble method is first verified on the flow in a square duct, where it correctly learns the underlying turbulence models from velocity data. Then the generalizability of the learned model is evaluated on a family of separated flows over periodic hills. It is demonstrated that the turbulence model learned in one flow can predict flows in similar configurations with varying slopes.

**Key words:** turbulence modelling, machine learning

† Email addresses for correspondence: hengxiao@vt.edu, hgw@lnm.imech.ac.cn

## 1. Introduction

Despite the growth of available computational resources and the development of high-fidelity methods, industrial computational fluid dynamics (CFD) simulations still rely predominantly on Reynolds-averaged Navier–Stokes (RANS) solvers with turbulence models. This is expected to remain so in the decades to come, particularly for outer loop applications such as design optimization and uncertainty quantification (Slotnick *et al.* 2014). Therefore, it is still of practical interest to develop more accurate and robust turbulence models.

Most of the models used currently are linear eddy viscosity models such as the $k$–$\varepsilon$ model (Launder & Sharma 1974) and Spalart–Allmaras model (Spalart & Allmaras 1992), which are based on two major assumptions (Pope 2000): (1) the weak equilibrium assumption, i.e. only the non-equilibrium in the magnitude of the Reynolds stress is accounted for through the transport equations, while its anisotropy is modelled based on local strain rate; and (2) the Boussinesq assumption, i.e. the Reynolds stress anisotropy is assumed to be aligned with the strain rate tensor. Reynolds stress transport models (also referred to as differential stress models) have been developed in the past few decades to address the shortcomings caused by the weak equilibrium assumption (Launder, Reece & Rodi 1975; Speziale, Sarkar & Gatski 1991; Eisfeld, Rumsey & Togiti 2016). As to the second assumption, various nonlinear eddy viscosity and explicit algebraic stress models have been developed (Spalart 2000; Wallin & Johansson 2000), and some have even achieved dramatic successes in specialized flows (e.g. those with secondary flows or rotation). However, these complex models face challenges from the lack of robustness, increased computational costs and implementation complexity, and the difficulty of generalizing to a broader range of flows. Consequently, turbulence modellers and CFD practitioners often face a compromise between predictive performance and practical usability (Xiao & Cinnella 2019).

In the past few years, data-driven methods have emerged as a promising alternative for developing more generalizable and robust turbulence models. For example, non-local models based on vector-cloud neural networks have been proposed to emulate Reynolds stress transport equations (Han, Zhou & Xiao 2022; Zhou, Han & Xiao 2022). While this line of research is still in an early stage, it has the potential of leading to more robust and flexible non-equilibrium Reynolds stress models without solving the tensorial transport equations. Alternatively, data-driven nonlinear eddy viscosity models have achieved much more success. Researchers have used machine learning to discover data-driven turbulence models or corrections thereto, which are nonlinear mappings from the strain rate and rotation rate to Reynolds stresses learned from data. Such functional mappings can be in the form of symbolic expressions (Weatheritt & Sandberg 2016; Schmelzer, Dwight & Cinnella 2020), tensor basis neural networks (Ling, Kurzawski & Templeton 2016), and random forests (Wang, Wu & Xiao 2017; Wu, Michelén-Ströfer & Xiao 2019*a*), among others. The data-driven nonlinear eddy viscosity models are a major improvement over their traditional counterparts in that they can leverage calibration data more systematically and explore a much larger functional space of stress–strain-rate mappings. However, they have some major shortcomings. First, as with their traditional counterparts, these data-driven models addressed only the Boussinesq assumption of the linear models as their strain–stress relations are still local, and thus they cannot address the weak equilibrium assumption described above. This is in contrast to the data-driven non-local Reynolds stress models (Han *et al.* 2022; Zhou *et al.* 2022), which emulate the Reynolds stress transport equations and fully non-equilibrium models. Second, the training of such models

often requires full-field Reynolds stresses (referred to as direct data hereafter), which are rarely available except from high-fidelity simulations such as direct numerical simulations (DNS) and wall-resolved large eddy simulations (LES) (Yang & Griffin 2021). This would inevitably constrain the training flows to those accessible for DNS and LES, i.e. flows with simple configurations at low Reynolds numbers. It is not clear whether the data-driven models trained with such data would be applicable to practical industrial flows. Finally, the training of data-driven models is often performed in an *a priori* manner, i.e. without involving RANS solvers in the training process. Consequently, the trained model may have poor predictions of the mean velocity in *a posteriori* tests where the trained turbulence model is coupled with the RANS solvers. This is caused by the inconsistency between the training and prediction environments (Duraisamy 2021). Specifically, even small errors in the Reynolds stress can be amplified dramatically in the predicted velocities due to the intrinsic ill-conditioning of the RANS operator (Wu *et al.* 2019*b*; Brener *et al.* 2021). Such ill-conditioning is particularly prominent in high-Reynolds-number flows; even an apparently simple flow such as a plane channel flow can be extremely ill-conditioned (Wu *et al.* 2019*b*). Such an inconsistency can be more severe in learning dynamic models such as subgrid-scale models of LES, since the training environment is static while the prediction environment is dynamic. Moreover, the model with the best *a posterior* performance may not necessarily excel in *a priori* evaluations (Park & Choi 2021). In view of the drawbacks in *a priori* training of turbulence models with direct data (Reynolds stress), it is desirable to leverage indirect observation data (e.g. sparse velocities and drag) to train data-driven turbulence models in the prediction environments by involving the RANS solvers in the training process. These indirect data are often available from experiments at high Reynolds numbers. Such a strategy is referred to as 'model-consistent learning' in the literature (Duraisamy 2021).

Model-consistent learning amounts to finding the turbulence model that, when embedded in the RANS solvers, produces outputs in the best agreement with the training data. Specifically, in incompressible flows, these outputs include the velocity and pressure as well as their post-processed or sparsely observed quantities. Assuming that the turbulence model is represented by a neural network to be trained with the stochastic gradient descent method, every iteration in the training process involves solving the RANS equations and finding the sensitivity of the discrepancy between the observed and predicted velocities with respect to the neural network weights. This is in stark contrast to the traditional method of training neural networks that learns from direct data (output of the neural network, i.e. Reynolds stresses in this case), where the gradients can be obtained directly from back-propagation. In model-consistent training, one typically uses adjoint solvers to obtain the RANS-solver-contributed gradient (i.e. the sensitivity of velocity with respect to the Reynolds stress), as the full model consists of both the neural network and the RANS solver (Holland, Baeder & Duraisamy 2019; Michelén-Ströfer & Xiao 2021). The adjoint sensitivity is then multiplied by the neural network gradient according to the chain rule to yield the full gradient. Similar efforts of combining adjoint solvers and neural network gradients have been made in learning subgrid-scale models in LES (MacArt, Sirignano & Freund 2021). These adjoint-based methods have been demonstrated to learn models with good posterior velocity predictions. Moreover, for turbulence models represented as symbolic expressions, model-consistent learning is performed similarly by combining the model with the RANS solver in the learning processes (Zhao *et al.* 2020; Saïdi *et al.* 2022), although the chain-rule-based gradient evaluation is no longer needed in gradient-free optimizations such as genetic optimization.

In view of the extra efforts in developing adjoint solvers, particularly for legacy codes and multi-physics coupled solvers, Michelén-Ströfer, Zhang & Xiao (2021*b*) explored ensemble-based gradient approximation as an alternative to the adjoint solver used in Michelén-Ströfer & Xiao (2021) to learn turbulence models from indirect data. Such a gradient is combined with that from the neural network via the chain rule, and then used in an explicit gradient-descent training. They found that the learned model was less accurate than that learned by using adjoint solvers in the prediction of Reynolds stress and velocity. This is not surprising, because the ensemble-based gradient approximation is less accurate than the analytic gradient from the adjoint solvers (Evensen 2018). Therefore, instead of using an ensemble to approximate gradients in optimization, it can be advantageous to use ensemble Kalman methods directly for training neural networks (Chen *et al.* 2019; Kovachki & Stuart 2019). This is because such ensemble methods do not merely perform explicit, first-order gradient-descent optimization as is typically done in neural network training (deep learning). Rather, they use implicitly the Hessian matrix (second-order gradient) along with the Jacobian (first-order gradient) to accelerate convergence. Indeed, ensemble-based learning has gained significant success recently (Schneider, Stuart & Wu 2020*a*,*b*), but the applications focused mostly on learning from direct data. They have not been used to learn from indirect data, where physical models such as RANS solvers become an integral part of the learning process.

In this work, we propose using an iterative ensemble Kalman method to train a neural-network-based turbulence model by using indirect observation data. To the authors' knowledge, this is the first such attempt in turbulence modelling. Moreover, in view of the strong nonlinearity of the problem, we adjust the step size adaptively in the learning process (Luo *et al.* 2015), which serves a similar purpose to that of the learning-rate scheduling in deep learning. Such an algorithmic modification is crucial for accelerating convergence and improving robustness of the learning, which can make an otherwise intractable learning problem with the adjoint method computationally feasible with the ensemble method. A comparison is performed between the present method and the continuous adjoint method based on our particular implementation (Michelén-Ströfer & Xiao 2021). We show that by incorporating Hessian information with adaptive stepping, the ensemble Kalman method exceeds the performance of the adjoint-based learning in both accuracy and robustness. Specifically, the present method successfully learned a generalizable nonlinear eddy viscosity model for the separated flows over periodic hills (§ 4), which the adjoint method was not able to achieve due to the lack of robustness. We emphasize that all these improvements are achieved at a much lower computational cost (measured in wall time) and with a significantly lower implementation effort compared to the adjoint method. Both methods used the same representation of Reynolds stresses based on the tensor basis neural network (Ling *et al.* 2016).

In summary, the present framework of ensemble-based learning from indirect data has three key advantages. First, compared to methods that learn from direct data, the present framework relaxes the data requirements and needs only the measurable flow quantities, e.g. sparse measurements of the mean velocities or integral quantities such as drag and lift, rather than full-field Reynolds stresses. Second, the model is trained in the prediction environment, thereby alleviating the ill-condition of the explicit data-driven RANS equation and avoiding the inconsistency between training and prediction. Finally, the ensemble method is non-intrusive and thus very straightforward to implement for any solvers. In particular, it does not require adjoint solvers, which allows different quantities to be used in the objective function without additional code redevelopments.

The rest of this paper is organized as follows. The architecture of the neural network and the model-consistent training algorithm are presented in § 2. The case set-up for testing the performance of the proposed non-intrusive model-consistent training workflow is detailed in § 3. The training results are presented and analysed in § 4. The parallelization and the flexibility of the proposed method are discussed in § 5. Finally, conclusions are provided in § 6.

## 2. Reynolds stress representation and model-consistent training

The objective is to develop a data-driven turbulence modelling framework that meets the following requirements.

(i) The Reynolds stress representation will be frame invariant and sufficiently flexible in expressive power to represent a wide range of flows.
(ii) The model will be trained in the prediction environment for robustness.
(iii) The model will be able to incorporate sparse and potentially noisy observation data as well as Reynolds stress data.

To this end, we choose the tensor basis neural networks (Ling *et al.* 2016) to represent the mapping from the mean velocities to the Reynolds stresses. This representation has the merits of the embedded Galilean invariance and the flexibility to model complicated nonlinear relationships. Furthermore, we use the ensemble Kalman method to learn the neural-network-based model in a non-intrusive, model-consistent manner.

The proposed workflow for training the tensor basis neural networks with indirect observation data is illustrated schematically in figure 1. Traditionally, ensemble Kalman methods have been used in data assimilation applications to infer the state of the system (e.g. velocities and pressures of a flow field). However, in our application, we aim to learn a turbulence model represented by a neural network. Therefore, the parameters (weight vector $\boldsymbol{w}$) of the network are the quantities to be inferred. The iterative ensemble Kalman method adopted for model learning consists of the following steps.

(i) Sample the parameters (neural network weight vector $\boldsymbol{w}$) based on the initial prior distribution (figure 1*a*). The initial parameters are obtained by pre-training based on a baseline model.
(ii) Construct the Reynolds stress field from the mean velocity field by evaluating the neural-network-based turbulence model (figure 1*b*). The initial velocity field is obtained from the prediction with the baseline model. For a given mean velocity field $\boldsymbol{u}(\boldsymbol{x})$, each of the sample $\boldsymbol{w}_j$ (with $j$ being the sample index) implies a different turbulence model and thus a different Reynolds stress field, leading to an ensemble of Reynolds stress fields in the whole computational domain.
(iii) Propagate each Reynolds stress field in the ensemble to velocity field by solving the RANS equations (figure 1*c*), based on which the observations can be obtained via post-processing (e.g. extracting velocities at specific points or integrating surface pressure to obtain drag).
(iv) Update the parameters (network weights $\boldsymbol{w}$) through statistical analysis of the predicted observable quantities (e.g. velocities or drag) and comparison with observation data (figure 1*d*).

Steps (ii)–(iv) are repeated until convergence is achieved. The implementation details are provided in Appendix A.

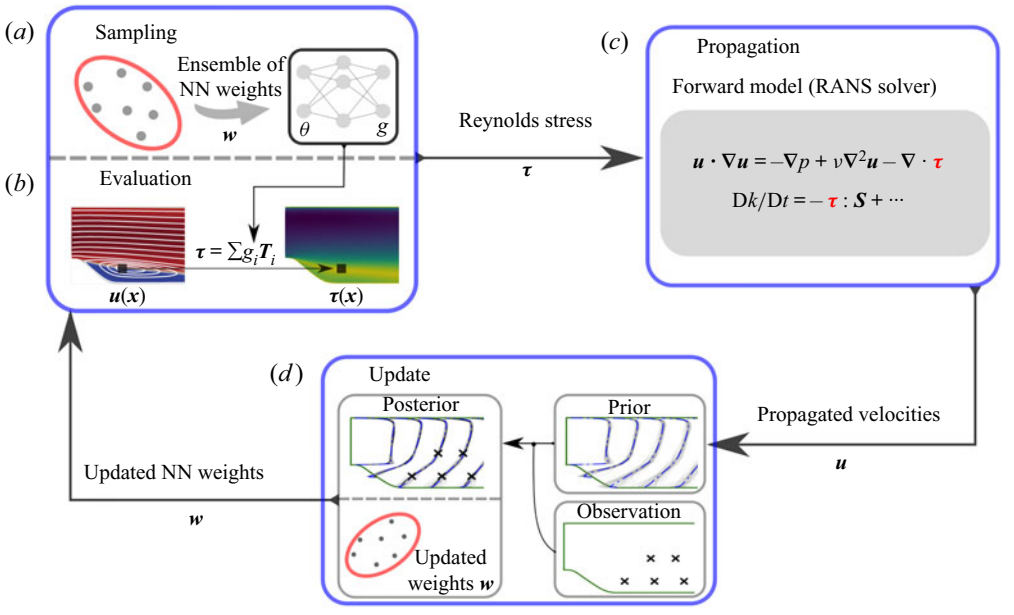



Figure 1. Schematic of the ensemble-based learning with sparse velocity data, consisting of the following four steps: (*a*) sample the weights of the tensor basis neural network (NN); (*b*) construct the Reynolds stress by evaluating the neural-network-based turbulence model; (*c*) propagate the constructed Reynolds stress tensor to velocity by solving the RANS equations; (*d*) update the neural network weights by incorporating observation data.

In this section, we introduce the Reynolds stress representation based on the tensor basis neural network and the ensemble-based learning algorithm. The latter is compared to other learning algorithms in the literature.

### 2.1. *Embedded neural network for Reynolds stress representation*

For constant-density incompressible turbulent flows, the mean flow can be described by the RANS equation

$$\left.\begin{aligned} \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0, \\ \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u} &= -\boldsymbol{\nabla}p + \nu\,\boldsymbol{\nabla}^2\boldsymbol{u} - \boldsymbol{\nabla}\cdot\boldsymbol{\tau}, \end{aligned}\right\} \tag{2.1}$$

where $p$ denotes mean pressure normalized by the constant flow density, and the Reynolds stress $\boldsymbol{\tau}$ indicates the effects of the small-scale turbulence on the mean flow quantities, which are required to be modelled. The Reynolds stress can be decomposed into a deviatoric part ($\boldsymbol{a}$) and a spherical part as

$$\boldsymbol{\tau} = \boldsymbol{a} + \tfrac{2}{3}k\boldsymbol{I}, \tag{2.2}$$

where $k$ is the turbulence kinetic energy, and $\boldsymbol{I}$ is the second-order identity tensor. Different strategies have been developed to represent the deviatoric part of the Reynolds stress, and here we use the tensor basis neural network (Ling *et al.* 2016).

The neural network represents the deviatoric part of Reynolds stress with the scalar invariants and the tensor bases of the turbulence field. Specifically, the neural network is used to represent the mapping between the scalar invariants and coefficients of the tensor bases. Further, the output of the neural network is combined with the tensor bases to

construct the Reynolds stress field such that the framework has the embedded Galilean invariance. The deviatoric part of the Reynolds stress ($\boldsymbol{a}$) can be constructed as (Pope 1975)

$$\boldsymbol{a} = 2k\sum_{i=1}^{10} g^{(i)}\boldsymbol{T}^{(i)}, \tag{2.3}$$

$$\text{with} \quad g^{(i)} = g^{(i)}(\theta_1, \ldots, \theta_5), \tag{2.4}$$

where $\boldsymbol{T}$ and $\boldsymbol{\theta}$ are the tensor basis and scalar invariant of the input tensors, and $\boldsymbol{g}$ is the scalar coefficient functions to be learned. There are 10 independent tensors that give the most general form of eddy viscosity. The first four tensors are given as

$$\left.\begin{aligned} \boldsymbol{T}^{(1)} = \boldsymbol{S}, \quad \boldsymbol{T}^{(2)} = \boldsymbol{SW} - \boldsymbol{WS} \\ \boldsymbol{T}^{(3)} = \boldsymbol{S}^2 - \tfrac{1}{3}\{\boldsymbol{S}^2\}\boldsymbol{I}, \quad \boldsymbol{T}^{(4)} = \boldsymbol{W}^2 - \tfrac{1}{3}\{\boldsymbol{W}^2\}\boldsymbol{I}, \end{aligned}\right\} \tag{2.5}$$

where the curly brackets $\{\cdot\}$ indicate the trace of a matrix. The first two scalar invariants are

$$\theta_1 = \{\boldsymbol{S}^2\} \quad \text{and} \quad \theta_2 = \{\boldsymbol{W}^2\}. \tag{2.6a,b}$$

Both the symmetric tensor $\boldsymbol{S}$ and the anti-symmetric tensor $\boldsymbol{W}$ are normalized by the turbulence time scale $k/\varepsilon$ as $\boldsymbol{S} = \frac{1}{2}k/\varepsilon[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathrm{T}}]$ and $\boldsymbol{W} = \frac{1}{2}k/\varepsilon[\nabla\boldsymbol{u} - (\nabla\boldsymbol{u})^{\mathrm{T}}]$. The time scale $k/\varepsilon$ is obtained from the turbulence quantities solved from the transport equations for turbulence kinetic energy $k$ and dissipation rate $\varepsilon$. For a two-dimensional flow, only two scalar invariants are non-zero, and the first three tensor bases are linearly independent (Pope 1975). Further, for incompressible flows, the components of the third tensor have $\boldsymbol{T}^{(3)}_{11} = \boldsymbol{T}^{(3)}_{22}$ and $\boldsymbol{T}^{(3)}_{12} = \boldsymbol{T}^{(3)}_{21} = 0$. Hence the third tensor basis can be incorporated into the pressure term in the RANS equation, leaving only two tensor functions and two scalar invariants. In the turbulence transport equation, the turbulence production term is modified to account for the expanded formulation of Reynolds stress $\mathcal{P} = -\boldsymbol{\tau} : \boldsymbol{S}$, where : denotes double contraction of tensors. For details of the implementation, readers are referred to Michelén-Ströfer & Xiao (2021). Note that the representation of the Reynolds stress is based on the following three hypotheses: (1) the Reynolds stress can be described locally with the scalar invariants and the independent tensors; (2) the coefficients of the tensor bases can be represented by a neural network with the scalar invariants as input features; (3) a universal model form exists for flows having similar distributions of scalar invariants in the feature space. Admittedly, the nonlinear eddy viscosity model is essentially still under the weak equilibrium assumption. Here, we choose the nonlinear eddy viscosity model as a base model, mainly due to the following considerations. First, it is a more general representation of the Reynolds stress tensor compared to the linear eddy viscosity model. It utilizes ten tensor bases formed by the strain-rate tensor and rotation-rate tensor to represent the Reynolds stress, while the linear eddy viscosity model uses only the first tensor basis. Second, the nonlinear eddy viscosity model uses uniform model inputs with Galilean invariance, i.e. scalar invariants, without requiring feature selections based on physical knowledge of specific flow applications. Finally, the model expresses the Reynolds stress in an algebraic form, and no additional transport equation is solved. From a practical perspective, the model is straightforward to implement and computationally more efficient than the Reynolds stress transport models.

In this work, the tensor basis neural network is embedded into the RANS equation during the training process. Specifically, the RANS equation is solved to propagate the

Reynolds stress to the velocity by coupling with the neural-network-based model, and the propagated velocity and the indirect observations are analysed to train the neural network through model learning algorithms. We use an ensemble Kalman method to train the neural-network-based turbulence model embedded in the RANS equations, which is elaborated in § 2.2. More detailed comparisons between the proposed method and other related schemes are presented in § 2.3.

### 2.2. *Ensemble-based model-consistent training*

The goal of the model-consistent training is to reduce the model prediction error by optimizing the weights $\boldsymbol{w}$ of the neural network. The corresponding cost function can be formulated as (Zhang, Michelén-Ströfer & Xiao 2020*a*)

$$J = \|\boldsymbol{w} - \boldsymbol{w}^0\|_P^2 + \|y - \mathcal{H}[\boldsymbol{w}]\|_R^2, \tag{2.7}$$

where $\|\cdot\|_A$ indicates weighted norm (defined as $\|\boldsymbol{v}\|_A^2 = \boldsymbol{v}^{\mathrm{T}} A^{-1} \boldsymbol{v}$ for a vector $\boldsymbol{v}$ with weight matrix $A$), $P$ is the model error covariance matrix indicating the uncertainties of the initial weights, $R$ is the observation error covariance matrix, and $y$ is the training data which are subjected to the Gaussian noise $\epsilon \sim \mathcal{N}(0, R)$. For simplicity, we introduce the operator $\mathcal{H}$, which is a composition of the RANS solver and the associated post-processing (observation). It maps the weights $\boldsymbol{w}$ to the observation space (e.g. velocity or drag coefficient). The first term in (2.7) is introduced to regularize the updated weights $\boldsymbol{w}$ by penalizing large deviations from their initial values $\boldsymbol{w}^0$. The second term describes the discrepancy between the model prediction $\mathcal{H}[\boldsymbol{w}]$ and the observation $y$. The training of the neural network is equivalent to minimization of the cost function (2.7) by optimizing the weights $\boldsymbol{w}$. Note that the cost function can be modified to include other observation quantities such as friction coefficient and transition location.

In this work, we use the iterative ensemble Kalman method with adaptive stepping (Luo *et al.* 2015) to train the neural network framework. This algorithm is a variant of the ensemble-based method where the observation error covariance matrix $R$ is inflated such that the step size is adjusted adaptively at each iteration step. The corresponding cost function involves the regularization based on the difference from the last iteration, i.e.

$$J = \|\boldsymbol{w}_j^{l+1} - \boldsymbol{w}_j^l\|_P^2 + \|y_j - \mathcal{H}[\boldsymbol{w}_j^l]\|_{\gamma R}^2, \tag{2.8}$$

where $l$ is the iteration index, $j$ is the sample index, and $\gamma$ is a scaling parameter. The weight-update scheme of the iterative ensemble Kalman method is formulated as

$$\boldsymbol{w}_j^{l+1} = \boldsymbol{w}_j^l + K\left(y_j - \mathcal{H}[\boldsymbol{w}_j^l]\right), \tag{2.9a}$$

$$\text{with} \quad K = S_w S_y^{\mathrm{T}} \left(S_y S_y^{\mathrm{T}} + \gamma^l R\right)^{-1}. \tag{2.9b}$$

The square-root matrices $S_w$ and $S_y$ can be estimated from the ensemble at each iteration. See step (vi) and (A1) of the detailed implementation in Appendix A.

Note that the Kalman gain matrix above has a slightly different form than the more common formulation $K = PH^{\mathrm{T}}(HPH^{\mathrm{T}} + \gamma^l R)^{-1}$. This is because we have written the terms associated with the model error covariance matrix $P$ by using the square-root matrix

$S_w$ and its projection $S_y$ to the observation space, i.e.

$$P = S_w S_w^{\mathrm{T}} \quad \text{and} \quad S_y = HS_w, \tag{2.10a,b}$$

where $H$ is the local gradient of the observation operator $\mathcal{H}$ with respect to the parameter $\boldsymbol{w}$. The equivalence between the two formulations is illustrated in Appendix B.

The Kalman gain matrix in (2.9*b*) contains implicitly the inverse of the approximated second-order derivatives (Hessian matrix) as well as the gradient (Jacobian) of the cost function (both with respect to the weights $\boldsymbol{w}$). This can be seen from the derivations presented in Appendix B. Including both the gradient and the Hessian information significantly accelerates the convergence of the iteration process and thus improves the learning efficiency. This is in stark contrast to using only the gradient in typical training procedures of deep learning. Moreover, this is done in ensemble Kalman methods economically, without significant overhead in computational costs or memory footprint.

The inflation parameter $\gamma^l$ in (2.9*b*) can be considered a coefficient for adjusting the relative weight between the prediction discrepancies and the regularization terms. As such, we let

$$\gamma^l = \beta^l \{S_y^l (S_y^l)^{\mathrm{T}}\}/\{R\}, \tag{2.11}$$

where $\beta^l$ is a scalar coefficient whose value also changes over the iteration process. The detailed algorithm for scheduling $\beta^l$ (and thus $\gamma^l$) is presented in step (vii) of the detailed implementation in Appendix A.

The ensemble-based method has the following three practical advantages. First, it produces an ensemble of weights of the neural network, based on which uncertainty quantification can be conducted for the model prediction similarly to the Bayesian neural network (Sun & Wang 2020). Second, unlike the adjoint-based method, the ensemble-based method is non-intrusive and derivative-free, which means that it can be applied to black-box systems without the need for modifying the underlying source code. This feature makes it convenient to implement the ensemble-based method in practice, and promotes the generalizability of the implemented ensemble-based method to different problems. Finally, to reduce the consumption of computer memory, commonly used training algorithms, such as stochastic gradient descent, typically involve the use of only gradients of an objective function to update the weights of a neural network, while the ensemble-based method incorporates the information of a low-rank approximated Hessian without a substantial increment of computer memory. Utilizing the Hessian information significantly improves convergence, as discussed above. In addition, the method can be used to train the model jointly with data from different flow configurations. In such scenarios, the observation vector and the corresponding error covariance matrix would contain different quantities, e.g. the velocity and drag coefficient. The ensemble-based learning method interacts with the prediction environment during the training process, which is similar to the reinforcement learning in this sense. However, the reinforcement learning usually learns a control policy for a dynamic scenario (e.g. Novati, de Laroussilhe & Koumoutsakos 2021; Bae & Koumoutsakos 2022), while the present work learns a closure model with supervised learning. Moreover, the reinforcement learning approach uses particular policy gradient algorithms to update the policy, while the ensemble Kalman method uses the ensemble-based gradient and Hessian to find the minimum of the underlying objective function.

An open-source platform OpenFOAM (The OpenFOAM Foundation 2021) is used in this work to solve the RANS equations with turbulence models. Specifically, the built-in solver *simpleFoam* is applied to solve the RANS equation coupling with the specialized

neural network model. Moreover, the DAFI code (Michelén-Ströfer, Zhang & Xiao 2021*a*) is used to implement the ensemble-based training algorithm. A fully connected neural network is used in this work, and the detailed architecture for each case will be explained later. The rectified linear unit (ReLU) activation function is used for the hidden layers, and the linear activation function is used for the output layer. The machine learning library TensorFlow (Abadi *et al.* 2015) is employed to construct the neural network. The code developed for this work is publicly available on Github (Zhang *et al.* 2022).

### 2.3. *Comparison to other learning methods*

Various approaches have been proposed for data-driven turbulence modelling, such as the direct training method (Ling *et al.* 2016), the adjoint-based differentiable method (Holland *et al.* 2019; MacArt *et al.* 2021; Michelén-Ströfer & Xiao 2021), the ensemble gradient method (Michelén-Ströfer *et al.* 2021*b*), and the ensemble Kalman inversion (Kovachki & Stuart 2019). Here, we present an algorithmic comparison of the proposed method with other model learning strategies in a unified perspective.

Conventional methods use the Reynolds stress of DNS to train the model in the *a priori* manner, with the goal of minimizing the discrepancy between the output of a neural network and the training data based on the back-propagation technique. This concept can be formulated as a corresponding minimization problem (with the proposed solution), as follows:

$$\left.\begin{aligned} \arg\min_{\boldsymbol{w}} J &= \|\boldsymbol{\tau}(\boldsymbol{w}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{W}}) - \boldsymbol{\tau}^{DNS}\|^2, \\ \boldsymbol{w}^{l+1} &= \boldsymbol{w}^{l} - \beta \frac{\partial \boldsymbol{\tau}(\boldsymbol{w}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{W}})}{\partial \boldsymbol{w}} \left[\boldsymbol{\tau}(\boldsymbol{w}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{W}}) - \boldsymbol{\tau}^{DNS}\right], \end{aligned}\right\} \tag{2.12}$$

where the input features $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{W}}$ are processed from the DNS results. Further, the trained neural network is coupled with the RANS solver for the posterior tests in similar configurations. It is obvious that inconsistency exists between the training and prediction environments. Specifically, during the training process, the model inputs are post-processed from the DNS data, while the learned model uses the RANS prediction to construct the input features. Besides, the training process aims to minimize the cost function associated with the Reynolds stress, while the prediction aims to achieve the least discrepancies in the velocity. This inconsistency would lead to unsatisfactory prediction due to the ill-conditioning issue of the RANS equation (Wu *et al.* 2019*b*). To tackle this problem, model-consistent training is required to construct the input features and the cost function with respect to more appropriate predicted quantities, e.g. the velocity.

For model-consistent training, the corresponding minimization problem (together with its solution) is changed to

$$\left.\begin{aligned} \arg\min_{\boldsymbol{w}} J &= \|\boldsymbol{u}^{DNS} - \boldsymbol{u}(\boldsymbol{\tau}(\boldsymbol{w}, \boldsymbol{S}, \boldsymbol{W}))\|^2, \\ \boldsymbol{w}^{l+1} &= \boldsymbol{w}^{l} - \beta \frac{\partial J}{\partial \boldsymbol{w}}, \end{aligned}\right\} \tag{2.13}$$

where the input features $\boldsymbol{S}$ and $\boldsymbol{W}$ are processed from the RANS prediction. Both the input feature and the objective function used for training are consistent with the prediction environment. Different approaches can be used to train the model, such as the adjoint-based differentiable method, the ensemble-based gradient method, and the ensemble Kalman inversion method. Specifically, the adjoint-based differentiable

framework (Michelén-Ströfer *et al.* 2021*b*) decomposes the gradient of the cost function into $\partial J/\partial \boldsymbol{\tau}$ and $\partial \boldsymbol{\tau}/\partial \boldsymbol{w}$ by using the chain rule. The weight-update scheme can be written as

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^{l} - \beta \frac{\partial J}{\partial \boldsymbol{\tau}} \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{w}}. \tag{2.14}$$

The gradient $\partial J/\partial \boldsymbol{\tau}$ is computed using the adjoint-based method, and the gradient $\partial \boldsymbol{\tau}/\partial \boldsymbol{w}$ is computed based on the back-propagation method. The ensemble-based gradient method applies the Monte Carlo technique to draw samples from a Gaussian distribution. Moreover, the data noise is taken into account by weighting the cost function with the observation error covariance matrix $R$. Further, the cross-covariance matrix computed by the ensemble-based method can be used to approximate the adjoint-based gradient as

$$\frac{\partial J}{\partial \boldsymbol{\tau}} \approx S_{\tau} S_{y}^{\mathrm{T}} R^{-1} \left(\mathcal{H}[\boldsymbol{w}] - y\right). \tag{2.15}$$

The above-mentioned training approach employs the readily available analytic gradient of the neural network based on the back-propagation method. Further, the gradient of the cost function can be constructed by coupling with adjoint- or ensemble-based sensitivity of the RANS equation.

The ensemble Kalman inversion method (Kovachki & Stuart 2019) adds a regularization term into the cost function and approximates the gradient of the cost function with respect to the weights of the neural network based on implicit linearization. The minimization problem and the corresponding weight-update scheme are

$$\left.\begin{aligned} \arg\min_{\boldsymbol{w}} J &= \|\boldsymbol{w}^{l+1} - \boldsymbol{w}^{l}\|_{P}^{2} + \|\boldsymbol{u}^{DNS} - \boldsymbol{u}\|_{R}^{2}, \\ \boldsymbol{w}_{j}^{l+1} &= \boldsymbol{w}_{j}^{l} + S_{w}^{l} \left(S_{y}^{l}\right)^{\mathrm{T}} \left(S_{y}^{l} \left(S_{y}^{l}\right)^{\mathrm{T}} + R\right)^{-1} \left(y_{j} - \mathcal{H}[\boldsymbol{w}^{l}]\right). \end{aligned}\right\} \tag{2.16}$$

Note that this method involves the Hessian of the cost function (Evensen 2018; Luo 2021) and provides quantified uncertainties based on Bayesian analysis (Zhang *et al.* 2020*b*). Similar to the ensemble gradient method, the ensemble Kalman inversion method also approximates the sensitivity of velocity to neural network weights based on the ensemble cross-covariance matrix, without involving the analytic gradient of the neural network. However, the ensemble Kalman inversion method includes the approximated Hessian in the weight-update scheme, which is missing in the ensemble gradient method. The present algorithm can be considered a variant of the ensemble Kalman inversion method, which inherits the advantages of ensemble-based methods in terms of non-intrusiveness and quantified uncertainty. Moreover, the present method adjusts the relative weights of the prediction discrepancy and the regularization terms at each iteration step, which helps to speed up the convergence of the iteration process and enhance the robustness of the weight-update scheme. For convenience of comparison, the training algorithms of different model-consistent data-driven turbulence modelling frameworks are summarized in table 1.

The performance of the aforementioned methods in two applications, i.e. flow in a square duct and flow over periodic hills, is summarized in table 2. The square duct case is a synthetic case to assess the capability of the methods in learning underlying model functions, where the prediction with Shih's quadratic model (Shih 1993) is used as the training data. For this reason, learning from direct data (referred to as the 'direct learning method' hereafter) can construct the synthetic model function accurately, and the

| Method | Cost function | Update scheme |
|---|---|---|
| Learning from direct data | $J = \|\boldsymbol{\tau}^{DNS} - \boldsymbol{\tau}\|^2$ | $\boldsymbol{w}^{l+1} = \boldsymbol{w}^l + \beta \dfrac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{w}}(\boldsymbol{\tau}^{DNS} - \boldsymbol{\tau})$ |
| Adjoint-based learning | $J = \|\boldsymbol{u}^{DNS} - \boldsymbol{u}\|^2$ | $\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \beta \dfrac{\partial J}{\partial \boldsymbol{\tau}} \dfrac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{w}}$ |
| Ensemble gradient learning | $J = \|\boldsymbol{u}^{DNS} - \boldsymbol{u}\|_R^2$ | $\boldsymbol{w}_j^{l+1} = \boldsymbol{w}_j^l + K(y_j - \mathcal{H}[\boldsymbol{w}_j^l]) \dfrac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{w}}$, with $K = S_\tau S_y^{\mathrm{T}} R^{-1}$ |
| Ensemble Kalman method with adaptive stepping (present framework) | $J = \|\boldsymbol{w}_j^{l+1} - \boldsymbol{w}_j^l\|_P^2 + \|\boldsymbol{u}^{DNS} - \boldsymbol{u}\|_{\gamma R}^2$ | $\boldsymbol{w}_j^{l+1} = \boldsymbol{w}_j^l + K(y_j - \mathcal{H}[\boldsymbol{w}_j^l])$, with $K = S_w S_y^{\mathrm{T}} \left(S_y S_y^{\mathrm{T}} + \gamma R\right)^{-1}$ |

Table 1. Summary of different approaches for learning turbulence models in terms of the cost function and update schemes. We compared the ensemble Kalman method with adaptive stepping (Luo *et al.* 2015; Kovachki & Stuart 2019) with other related methods, including learning from direct data, i.e. the Reynolds stresses (Ling *et al.* 2016), adjoint-based learning (Holland *et al.* 2019; MacArt *et al.* 2021; Michelén-Ströfer & Xiao 2021) and ensemble gradient learning (Michelén-Ströfer *et al.* 2021*b*). The DNS mean velocities are used as example indirect data.

results are omitted for brevity. We present the results of the direct learning method for the periodic hill case in § 4.2, where the DNS data are used as the training data. The direct learning method is able to learn a model that improves the estimation of both velocity and Reynolds stress. However, when generalized to configurations with varying slopes, the learned model lacks robustness and leads to large discrepancies, as shown in figure 12. The adjoint-based learning method reconstructs accurately both the velocity and Reynolds stress fields in the square duct case, as shown in § 4.1. However, the method failed to learn a nonlinear eddy viscosity model in the periodic hill case – it diverged during the training as reported in Michelén-Ströfer & Xiao (2021). The ensemble gradient method was not able to recover the underlying model function in the synthetic square duct case (Michelén-Ströfer *et al.* 2021*b*), and also diverged in the periodic hill case. In contrast, the present method is capable of learning the functional mapping in both cases. Moreover, the learned model is generalized well to similar configurations with varying slopes, as shown in figure 12.

## 3. Case set-up

We use two test cases to show the performance of the proposed method for learning turbulence models: (1) flow in a square duct, and (2) separated flows over periodic hills. Both are classical test cases that are well-known to be challenging for linear eddy viscosity models (Xiao & Cinnella 2019). We aim to learn neural-network-represented nonlinear eddy viscosity models from velocity data by using the ensemble method. The learned models are evaluated by comparing to the ground truth for the square duct case and assessing its generalization performance in the separated flows over periodic hills. The results are also compared to those of the adjoint-based method. Details of the case set-up are discussed below.

### 3.1. *Secondary flows in a square duct*

The first case is the flow in a square duct, where the linear eddy viscosity model is not able to capture the in-plane secondary flow. The nonlinear eddy viscosity model, e.g. Shih's

| Method | Square duct (Learn synthetic model) | Periodic hill (Learn general nonlinear model) |
|---|---|---|
| Learning from direct data | — | *Poor* (see figure 12; learned nonlinear model; poor generalization) |
| Adjoint-based learning | *Good* (see figure 6; learned functional mapping) | *Diverged* (Only learned linear model) |
| Ensemble gradient learning | *Poor* (Failed to learn functional mapping) | *Diverged* |
| Ensemble Kalman method with adaptive stepping (present framework) | *Good* (see figure 6; learned functional mapping) | *Good* (see figure 12; learned nonlinear model; generalized well) |

Table 2. Summary of the performance of different approaches for learning turbulence models in two different cases, i.e. flow in a square duct and flow over periodic hills. We compare the present method (Luo *et al.* 2015; Kovachki & Stuart 2019) with other related methods, including learning from direct data (Ling *et al.* 2016), adjoint-based learning (Holland *et al.* 2019; MacArt *et al.* 2021; Michelén-Ströfer & Xiao 2021) and ensemble gradient learning (Michelén-Ströfer *et al.* 2021*b*). The square duct case uses the prediction from Shih's quadratic model (Shih 1993) as training data, while the periodic hill case uses the DNS results as training data.

quadratic model (Shih 1993), is able to simulate the secondary flows. Furthermore, Shih's quadratic model provides an explicit formula for the mapping between the scalar invariant $\boldsymbol{\theta}$ and the function $\boldsymbol{g}$, which serves as an ideal benchmark for evaluating the accuracy of the trained model functions. In Shih's quadratic model, the $\boldsymbol{g}$ function of the scalar invariant $\boldsymbol{\theta}$ is written as

$$\left.\begin{aligned} g^{(1)}(\theta_1,\theta_2) &= \frac{-2/3}{1.25+\sqrt{2\theta_1}+0.9\sqrt{-2\theta_2}},\\ g^{(2)}(\theta_1,\theta_2) &= \frac{7.5}{1000+\left(\sqrt{2\theta_1}\right)^3},\\ g^{(3)}(\theta_1,\theta_2) &= \frac{1.5}{1000+\left(\sqrt{2\theta_1}\right)^3},\\ g^{(4)}(\theta_1,\theta_2) &= \frac{-9.5}{1000+\left(\sqrt{2\theta_1}\right)^3}. \end{aligned}\right\} \tag{3.1}$$

Hence we use the velocity results from Shih's quadratic model as the synthetic truth and show that the method is able to reveal the underlying relationship between the scalar invariant and the tensor basis when the model exists in the form of the tensor bases. Moreover, we aim to compare the adjoint-based and the present ensemble-based methods in terms of the training accuracy and efficiency in this case.

The flow in a square duct is fully developed, and only one cell is used in the streamwise direction. Moreover, one-quarter of the domain is used due to the symmetry, and the mesh grid is $50 \times 50$. As for the architecture of the neural network in this case, two scalar invariants are used as input features, and four $g$ functions $g^{(1-4)}$ are used in the output layer. The input features of the synthetic truth are shown in figure 2. Since the streamwise velocity $u_x$ is dominant, the first two scalar invariants are approximately equal

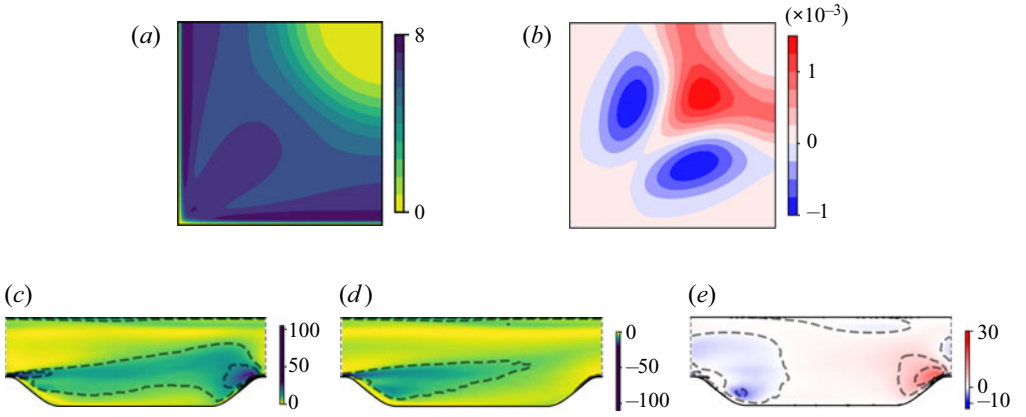



Figure 2. Contour plots of input features of the reference data for the square duct case and periodic hill case. (*a*) Square duct, $\theta_1$. (*b*) Square duct, $|\theta_1| - |\theta_2|$. (*c*) Periodic hills, $\theta_1$. (*d*) Periodic hills, $\theta_2$. (*e*) Periodic hills, $|\theta_1| - |\theta_2|$.

in magnitude but with opposite signs. The slight difference between the scalar invariants $\theta_1$ and $\theta_2$ is caused by the secondary flow in the plane. We also provide the plot of $|\theta_1| - |\theta_2|$, which indicates the relative importance of the strain rate and the vorticity. The streamwise velocity gradient is relatively small near the centre of the duct, leading to the negligible scalar invariant $\theta_1$. Moreover, the shear strain rate is dominant near the duct centre, while there is a pair of vortexes indicating the strong rotation rate. Also, it can be seen that the range of the input features is from 0 to approximately 7. We draw 50 samples of the neural network weights in this case. In the neural network, we use 2 hidden layers with five neurons per layer. A sensitivity study of the training algorithm to the neural network architecture and the observation data is provided in Appendix C.

### 3.2. *Separated flow over periodic hills*

The flow over periodic hills is a canonical separated flow for the numerical investigation of turbulence models. There is no ground truth for the model function that is able to capture the flow characteristics accurately. Here, we use the DNS results (Xiao *et al.* 2020) as the training data, and learn the neural-network-based model by using the ensemble-based method. Further, we validate the generalizability of the learned model in similar configurations with varying slopes (Xiao *et al.* 2020). Specifically, the hill geometry is parametrized with the slope coefficient $\alpha$. The separation extent decreases as the slope $\alpha$ increases from 0.5 to 1.5. The case with slope parameter $\alpha = 1$ is used as the training case, and the cases with other slopes, $\alpha = 0.5, 0.8, 1.2, 1.5$, are used to test the generalizability of the learned model in the scenarios having different levels of flow separation. The mesh is set as 149 cells in the streamwise direction, and 99 cells in the normal direction after grid-independence tests. We use the $k$–$\varepsilon$ model (Launder & Sharma 1974) as the baseline model. The model learned from direct data is also provided for comparison. The implementation of the direct learning method is illustrated in Appendix D.

For the two-dimensional incompressible flow, there are only the first two scalar invariants and independent tensors after merging the third tensor basis into the pressure term in the RANS equation (Michelén-Ströfer & Xiao 2021). The input features of the

| Cases | Flow configuration | Training data |
|---|---|---|
| Case 1 | Square duct | Shih's quadratic model |
| Case 2 | Periodic hills ($\alpha = 0.5, 0.8, 1.0, 1.2, 1.5$) | DNS (Xiao *et al.* 2020) |

Table 3. Summary of the configurations and the training data.

DNS data are shown in figure 2, scaled with the RANS predicted time scale. The plot of the first scalar invariant $\theta_1$ indicates the large strain rate in the free shear layer and the windward side of the hill. The second scalar invariant $\theta_2$ shows the vorticity mainly in the flow separation region at the leeward side of the hill. From the plot of $|\theta_1| - |\theta_2|$, it can be seen that the magnitude of the first two scalars is equivalent in most areas. The strong vorticity in the downhill region is caused by the flow separation, while near the uphill region, the shear strain rate is dominant due to the channel contraction. Compared to the square duct case, the separated flow over periodic hills has a wider range in the magnitude of the input features, which is from 0 to about 100. That is because in the square duct case, the magnitude of the scalar invariant is determined mainly by the streamwise velocity $u_x$, while in the periodic hill case, both $u_x$ and $u_y$ have considerable effects on the input features. Moreover, the magnitude of the time scale in the periodic hill is much larger than that in the square duct flow. Concretely, the maximum value for the periodic hill case is about 490, while that for the square duct case is about 10. Hence we use a deeper neural network of 10 hidden layers with 10 neurons per layer compared to the square duct case based on the sensitivity analysis of the neural network architecture as shown in Appendix C. We draw 50 samples of the neural network weights in this case. The training data set is summarized in table 3.

## 4. Results

### 4.1. *Flow in a square duct: learning underlying closure functions*

We first use the proposed ensemble-based method to train the turbulence model for flows in a square duct, and the results show that the predicted Reynolds stress has a good agreement with the synthetic ground truth (3.1). Plots of the velocity and the Reynolds stress are presented in figures 3 and 4, with comparison to the adjoint-based method and the ground truth. The contour lines for $u_y$ are indicated in the velocity vector plot to clearly show similar patterns among the ground truth, the adjoint-based method and the ensemble-based method. The contour plots of the Reynolds stress in $\tau_{xy}$, $\tau_{yz}$, and $\tau_{yy}$ are used to demonstrate the ability of the ensemble-based method in discovering the underlying Reynolds stress model given velocity data. The in-plane velocity is driven by the Reynolds normal stresses imbalance $\tau_{yy} - \tau_{zz}$, which is evident from the vorticity transport equation (Launder & Sandham 2002). As such, the imbalance $\tau_{yy} - \tau_{zz}$ is also presented in figure 4, demonstrating that the Reynolds stress field is learned accurately from the in-plane velocities. The learned model with the proposed method achieves results in both the velocity and Reynolds stress similar to those of the adjoint-based method. The error contours are provided to show the error distribution of the adjoint-based and ensemble-based methods in the estimation of velocity and Reynolds stress. It is noticeable that the adjoint-based method can achieve better velocity estimation than the ensemble-based method. As for the Reynolds stress, the adjoint-based and ensemble-based

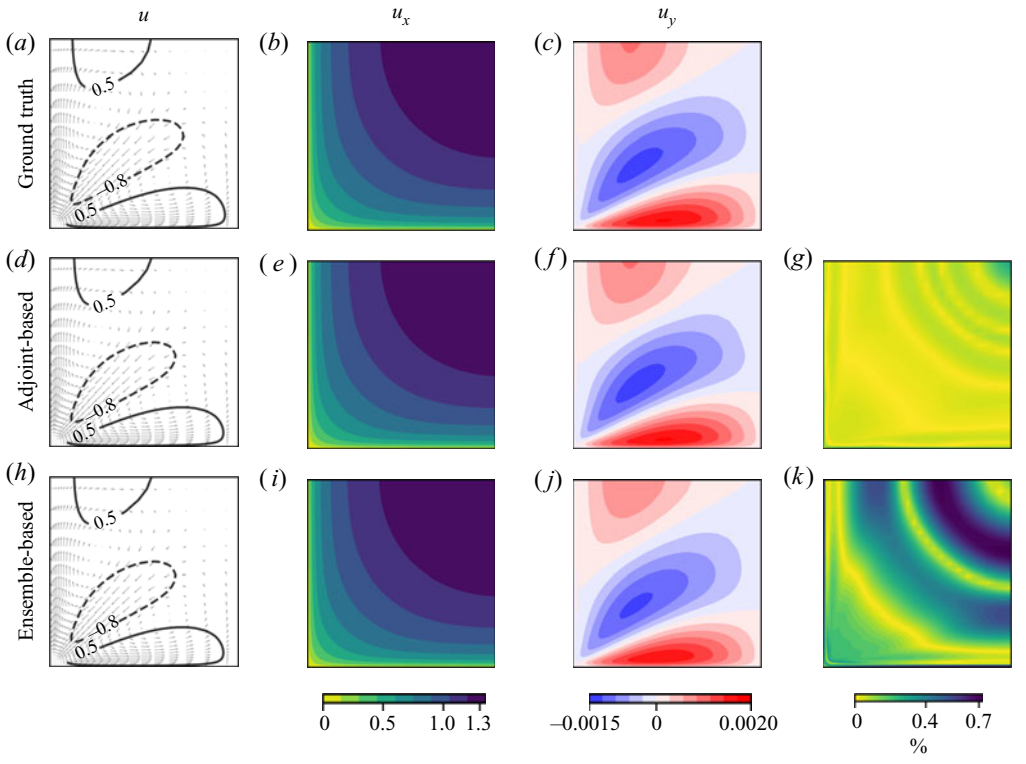



Figure 3. Plots of the velocity vector and components $u_x$ and $u_y$ in the square duct predicted from the models learned by the adjoint (*d–g*) and ensemble (*h–k*) methods, compared against the ground truth (*a–c*). The velocity vectors are plotted along with contours of the in-plane velocity $u_y$ scaled by a factor of 1000. The error contour is plotted based on $\|\boldsymbol{u} - \boldsymbol{u}^{truth}\|$ normalized by the maximum magnitude of $\boldsymbol{u}^{truth}$.

methods lead to similar results. It is noted that in this case, the entire field is used as the training data. By using fewer observations, e.g. only velocity data on the anti-diagonal line (upper right corner to lower left corner), the full velocity field can also be recovered, and the Reynolds stresses are learned correctly, but the errors are larger, especially in velocity. This is presented in Appendix C. The results demonstrate that the proposed method is able to learn the underlying turbulence model, which in turn provides good estimations of velocities and Reynolds stresses.

To show clearly the performance of the trained model, we provide the error in the estimation of velocity and Reynolds stress. The error over the computational domain is defined as

$$\mathcal{E}(\boldsymbol{q}) = \frac{\|\boldsymbol{q}^{predict} - \boldsymbol{q}^{truth}\|}{\|\boldsymbol{q}^{truth}\|}. \tag{4.1}$$

The comparison between the adjoint- and ensemble-based methods in the error of velocity and Reynolds stress, as well as the training efficiency, is provided in table 4. The results confirm that both adjoint- and ensemble-based methods are able to achieve satisfactory agreements in the velocities and to predict the Reynolds stresses well. By contrast, the adjoint-based method provides slightly better estimation than the ensemble-based method. Specifically, the errors in velocity and Reynolds stress with the adjoint-based method are

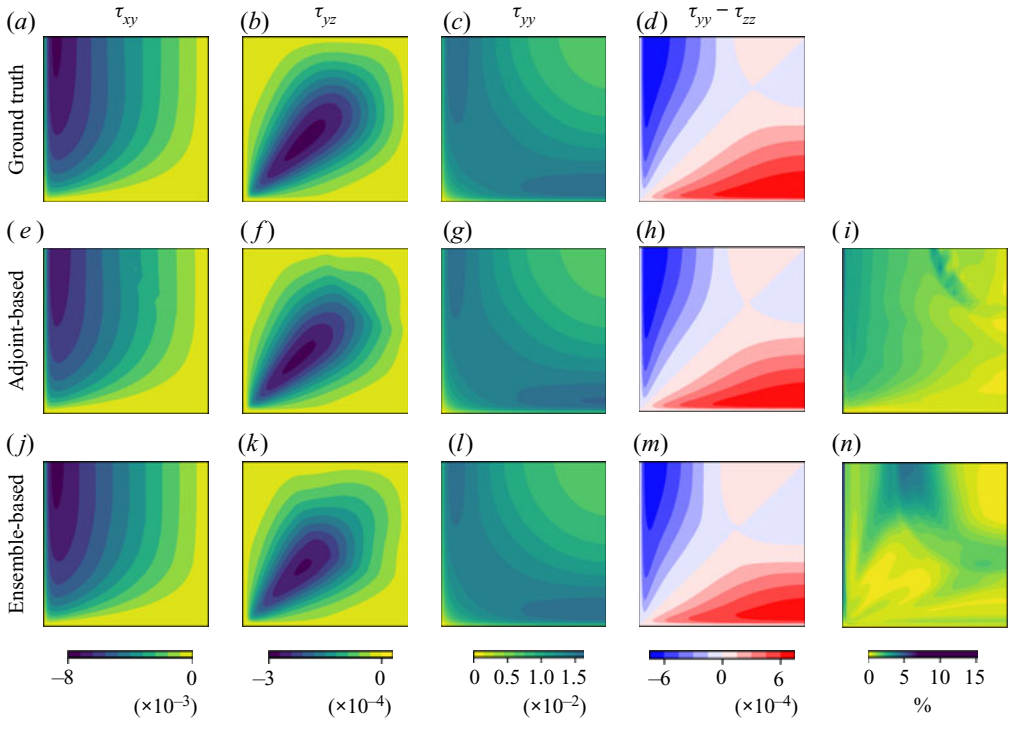



Figure 4. Plots of Reynolds shear stresses $\tau_{xy}$ and $\tau_{yz}$, normal stress $\tau_{yy}$, and normal stresses imbalance $\tau_{yy} - \tau_{zz}$, in the square duct predicted from the models learned by the adjoint (*e–i*) and ensemble (*j–n*) methods, compared against the ground truth (*a–d*). The error contour is plotted based on $\|\boldsymbol{\tau} - \boldsymbol{\tau}^{truth}\|$ normalized by the maximum magnitude of $\boldsymbol{\tau}^{truth}$.

| Method | $\mathcal{E}(\boldsymbol{u})$ | $\mathcal{E}(\boldsymbol{\tau})$ | Total steps | Wall time | Steps ($\mathcal{E}(\boldsymbol{u}) < 0.005$) | Wall time ($\mathcal{E}(\boldsymbol{u}) < 0.005$) |
|---|---|---|---|---|---|---|
| Adjoint-based | 0.1 % | 4.5 % | 1000 | 133 h | 238 | 32 h |
| Ensemble-based | 0.47 % | 5.8 % | 50 | 3.6 h | 8 | 0.6 h |

Table 4. Comparison of the estimation error and time cost between adjoint-based and ensemble-based learning.

0.1 % and 4.5 %, respectively, while those for the ensemble-based method are 0.47 % and 5.8 %, respectively.

As for the training efficiency, the adjoint-based method is more time-consuming compared to the ensemble-based method, as shown in table 4. Specifically, the adjoint-based method requires approximately 1000 iterations, which significantly increase the wall time to about 133 h in this case. In contrast, the ensemble-based method is efficient to obtain comparable results within 3.6 h. To achieve error reduction $\mathcal{E}(\boldsymbol{u}) < 0.005$, the adjoint-based method requires 238 steps and wall time 32 h, while the ensemble-based method can reach the same error within only 0.6 h. That is likely due to the use of Hessian information and the covariance inflation factor $\gamma$, which adjusts dynamically the relative weight of the cost function to accelerate the convergence (Nocedal &

Wright 2006). Here, we emphasize that the adjoint-based learning for comparison is based on our particular implementation of the continuous adjoint-based method (Othmer 2008). The used adjoint solver is publicly available in the Github repository (Zhang *et al.* 2022). Recent developments of the adjoint-based method, such as the online adjoint-based method (Sirignano & Spiliopoulos 2022), would have significant potential to improve the efficiency of the adjoint-based learning method. However, the present work aims to introduce the ensemble Kalman method for learning turbulence models, and comprehensive comparisons with the state-of-the-art adjoint-based method are out of the scope of this paper. It is also noted that this work focuses mainly on the steady-state RANS problem where the data size is small. In scenarios with large data sets such as unsteady three-dimensional flow fields, the present algorithm would be computationally expensive compared to the adjoint-based method, since the update scheme requires the inversion of a matrix with rank the dimensionality of observation data. In view of this limitation, dimension reduction techniques such as the truncated singular value decomposition are usually incorporated into the ensemble-based method (Evensen 2009), enabling it to handle large data sets. This strategy has been applied widely in large-scale reservoir applications (Chen & Oliver 2013; Luo, Bhakta & Naevdal 2018).

We further show the good reconstruction in the scalar invariant $\theta_1$ and $|\theta_1| - |\theta_2|$ with the ensemble-based method compared to the ground truth. The contour plots of the scalar invariant are presented in figure 5. The predicted scalar invariant with the learned model agrees well with the ground truth. The difference between the initial and the truth is mainly due to the in-plane secondary flow that cannot be captured by the linear eddy viscosity model. With the learned models, the flow field in the $y$–$z$ plane is well predicted, which further improves the estimate of the scalar invariant. It is observed that slight differences exist near the duct centre. In that region, there are mainly small values of the scalar invariant $\boldsymbol{\theta}$, due to the negligible streamwise velocity gradient. Additionally, we provide the estimated scalar invariant compared to the ground truth, which shows clearly the good agreements between the estimation and the truth. The probability density function of the scalar invariant $\boldsymbol{\theta}$ is also plotted in figure 5, showing the significantly small probability for $\boldsymbol{\theta}$ less than about 5. The 30 % quantile is located at approximately 5.1, indicating that only 30 % of the cells in the domain have $\theta_1$ smaller than this value.

The learned functional mappings between the scalar invariant $\boldsymbol{\theta}$ and the tensor basis coefficient $\boldsymbol{g}$ also have a good agreement with the ground truth. This is illustrated in figure 6. Since the two invariants are linearly correlated ($\theta_1 \approx -\theta_2$), we show only the plot of the mapping from the scalar invariant $\theta_1$ to the coefficients $\boldsymbol{g}$. It can be seen that the learned mapping can have a good agreement with the ground truth (the $\boldsymbol{g}(\boldsymbol{\theta})$ in (3.1)) implied by Shih's quadratic model. Although the combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$ is close to the ground truth, the components $g^{(2)-(4)}$ have significant discrepancies. That is because, in the duct flow, the in-plane velocity is affected by the linear combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$ of the $\boldsymbol{g}$ functions. We note that relatively large differences exist in the region with small values of $\theta_1$, particularly for the combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$. That is because the velocity is affected by the product of the $\boldsymbol{g}$ function and the tensor bases $\boldsymbol{T}$. In the region with small $\theta_1$ (near the centre of the duct), the magnitudes of the tensor bases $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$ (even after normalization with $k/\varepsilon$) are small, and thus the velocities are no longer sensitive to the $\boldsymbol{g}$ functions. Moreover, small values of $\theta_1$ are represented by only a small number of cells in the domain, which is evident from figure 5(*b*). Only 30 % of the cells in the domain have $\theta_1$ smaller than around 5, which is likely responsible for

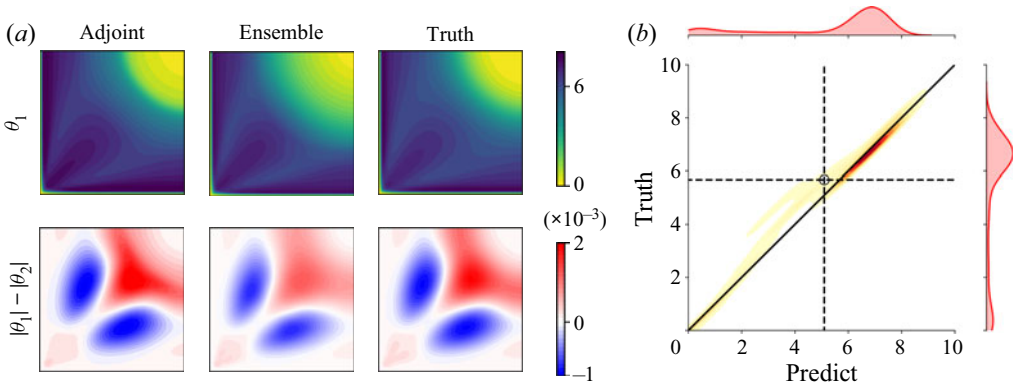



Figure 5. (*a*) Contours of scalar invariant $\theta_1$ and $|\theta_1| - |\theta_2|$, with comparison among the adjoint-based learned model, the ensemble-based learned model, and the truth. (*b*) Kernel density plot of $\theta_1$ from the truth and the estimation with the ensemble-based learned model. The circle indicates the 30 % quantile (i.e. 30 % of the cells have $\theta_1$ smaller than this value). The probability densities of the truth and the estimation are plotted on the margins.

the large discrepancies of the $\boldsymbol{g}$ functions in the range $\theta_1 < 5$. This lack of representation makes it difficult to learn the underlying mapping in the region with small $\theta_1$. However, we note that the ensemble-based method achieves results (albeit with errors of opposite signs) that are qualitatively similar to those from the adjoint-based method in the functional mapping. This suggests that the bottleneck for learning the complete mapping lies in the intrinsic ill-conditioning of the problem (insensitivity to small $\theta_1$ magnitudes) rather than the lack of analytic gradient. Meanwhile, the ill-conditioning problem may be remedied by learning from several flows with a wider range of $\boldsymbol{\theta}$.

### 4.2. *Flow over periodic hills: generalizability test to varying slopes*

The proposed method is further used to train the neural-network-based model for the flows over periodic hills. The flow with slope $\alpha = 1$ is used to train the model. The ensemble-based method is capable of reconstructing the flow field accurately in this case. This is shown in figure 7, where the velocity contour is provided with comparison to the results of the direct learning method and the DNS. It can be seen that the flow characteristics are well captured by minimizing the discrepancies between the velocity estimation and the given data. It is noted that only four velocity profiles, at $x/H = 1$, 3, 5 and 7, are used to achieve the improved reconstruction of the entire field. The separation bubbles with the direct learned model, the ensemble-based learned model and the truth are also provided in figure 7. The learned models with the direct learning method and the ensemble-based method both can well capture the bubble structure.

To show clearly the improvement in the velocity estimation, we present the comparison results along profiles in figures 8(*a*) and 8(*b*). The velocity profiles are improved significantly compared to the $k$–$\varepsilon$ model predictions. In particular, in the separation region, both velocities $u_x$ and $u_y$ are well predicted in good agreement with the DNS results. The comparison in the friction coefficient along the bottom wall is plotted in figure 8(*c*). The position of the reattachment point with the $k$–$\varepsilon$ model deviates substantially from the DNS. In contrast, the ensemble-based learned model can significantly improve the friction coefficient estimation, and especially the reattachment point is very close to the truth. The results with the direct learned model are provided for comparison, and the

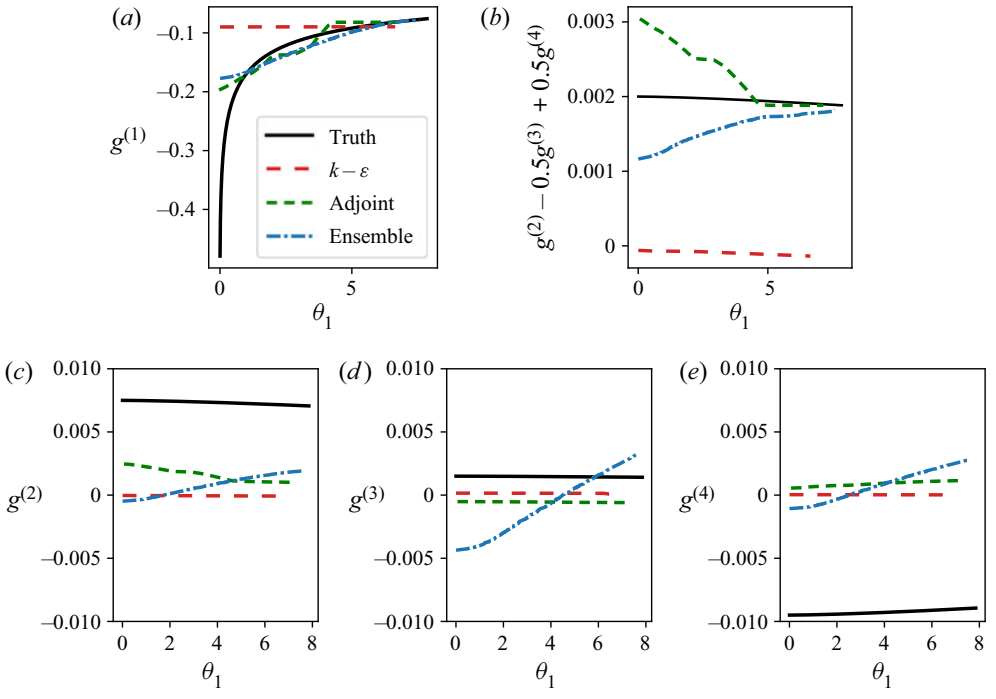



Figure 6. Comparison plots of the functional mapping between the scalar invariant and the tensor coefficient $g$ among the truth, the baseline $k$–$\varepsilon$ model, and the models learned with adjoint and ensemble methods: (*a*) $g^{(1)}$, (*b*) $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$, (*c*) $g^{(2)}$, (*d*) $g^{(3)}$, and (*e*) $g^{(4)}$.

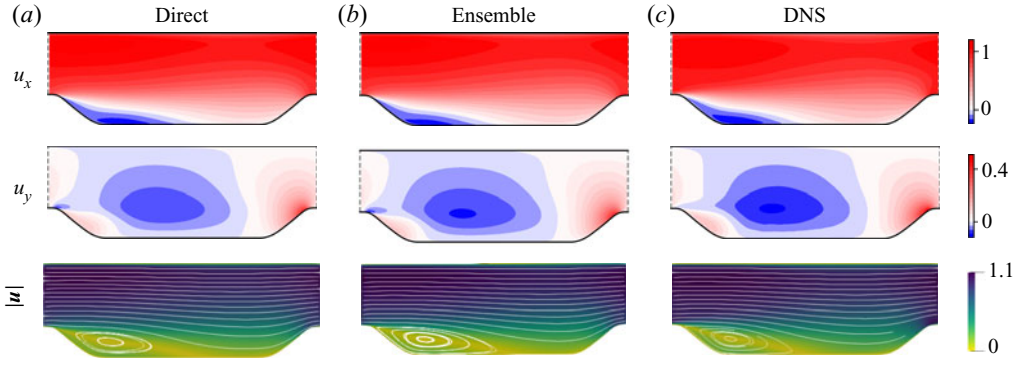



Figure 7. Contour plots of the velocity with the direct learned model, the ensemble-based learned model, and DNS for the periodic hill case. Note that the plots are from in-sample tests.

propagated velocity and friction coefficient are also improved noticeably compared to the $k$–$\varepsilon$ model. The estimation error and training efficiency of the direct learning and ensemble-based methods are shown in table 5. The two methods achieve similar velocity estimation, while the Reynolds stress with the direct learning method is slightly better than the ensemble-based method, due to the use of direct data. As for the training efficiency, the cost of the direct learning method is around 0.2 h, which is significantly lower than the present method (4.9 h), mainly because the CFD solver is not involved in the learning process.

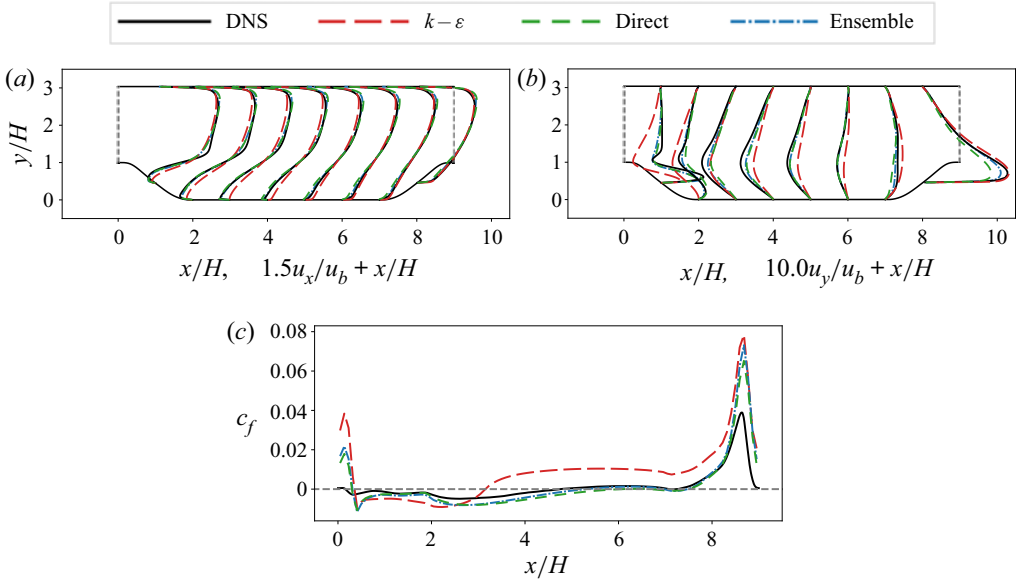



Figure 8. Comparison of velocity and the friction coefficient $c_f$ at the bottom wall along profiles among the $k$–$\varepsilon$ model, the direct learned model, the ensemble-based learned model, and the DNS for the periodic hill case. Note that the plots are from in-sample tests. (*a*) Horizontal velocity, $u_x$. (*b*) Vertical velocity, $u_y$. (*c*) Friction coefficient, $c_f$.

| Method | $\mathcal{E}(\boldsymbol{u})$ | $\mathcal{E}(\boldsymbol{\tau})$ | Total steps | Wall time |
|---|---|---|---|---|
| Direct learning | 6.6 % | 40.0 % | 10 000 | 0.2 h |
| Ensemble-based | 6.4 % | 44.0 % | 50 | 4.9 h |

Table 5. Comparison of the estimation error and time cost between the direct and ensemble-based learning methods for the periodic hill case.

The learned model has good predictive performance in capturing flow features for $\theta_1$ and $\theta_2$ at intermediate or small magnitudes. This is illustrated in figure 9(*a*), where the comparison of scalar invariants among the direct learned model, the ensemble-based learned model and the DNS data is presented. The scalar invariants $\boldsymbol{\theta}$ from both the direct learned model and the ensemble-based learned model exhibit patterns similar to those of the DNS data, while the noticeable difference exists mainly in $\boldsymbol{\theta}$ with large magnitudes around the separation point. This difference is likely due to the fact that the velocity data near the separation point are not used to train the model. Specifically, in this case the used data are distributed along four profiles, i.e. $x/H = 1, 3, 5$ and 7, which are away from the separation point. It can be seen from figure 9(*a*) that the magnitude of scalar invariants around the separation point significantly exceeds that of the training data. Hence the learned model with these data achieves only limited improvements in that region. To further improve the model estimation, additional data around the separation point should be used for training. It is noted that the training data, if positioned close spatially, may lead to poor training performance because the correlation among the observation errors is neglected. Specifically, the observation error includes the measurement and process errors in the ensemble Kalman method. The measurement error can be negligible for the

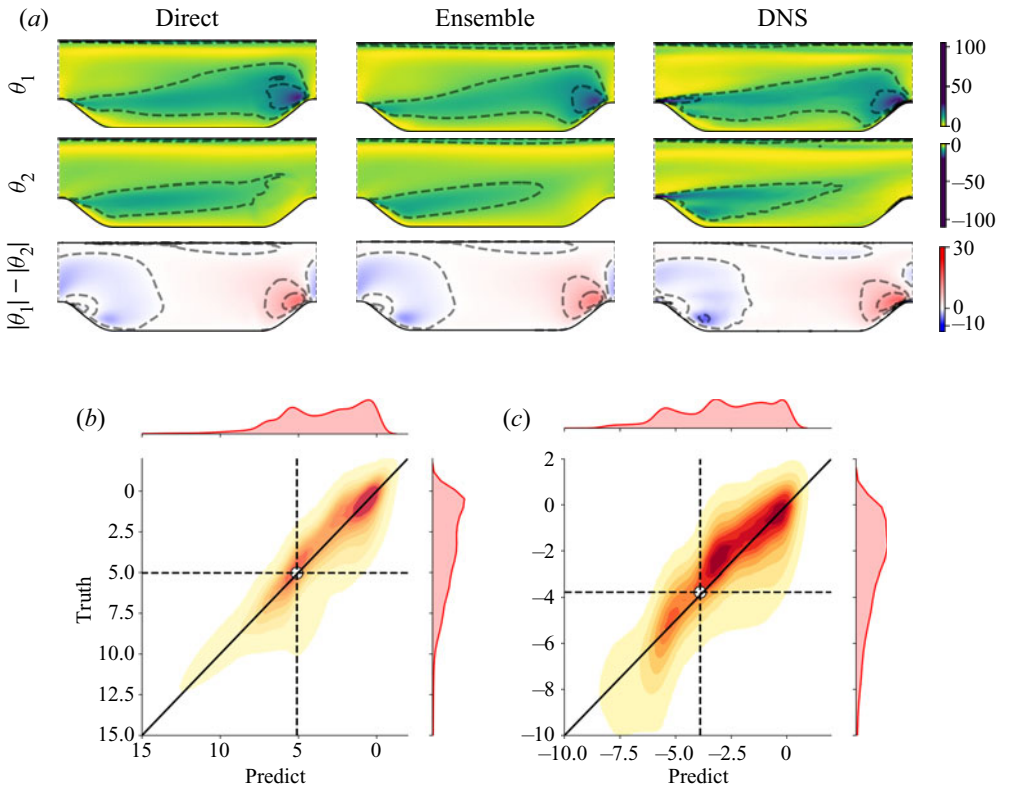



Figure 9. (*a*) Contour plots of $\theta_1$, $\theta_2$ and $|\theta_1| - |\theta_2|$, with comparison among the direct learned model, the ensemble-based learned model and the DNS. (*b*,*c*) Kernel density plots of $\theta_1$ and $\theta_2$ from the truth and the model estimation for periodic hill case, respectively. The round circles in (*b*,*c*) indicate the values of the 30 % quantiles (i.e. 30 % of the cells have $\boldsymbol{\theta}$ larger than this value in the magnitude). The probability densities of the truth and the model estimation are plotted on the margins.

DNS data, while the process error is significant in this case since it includes the intrinsic discrepancy between the RANS simulation and DNS. Such error correlation is difficult to estimate and often neglected, as in this work. However, the errors of spatially close data would have relatively strong correlations with each other, particularly in the streamwise direction due to the advection effects. As such, the neglected correlation information could deteriorate the training performance. Alternatively, one can place sparse data at particular positions, e.g. the separation point in this case, but the specific position is often not known *a priori*. Hence we suggest positioning training data evenly with a distance of more than one correlation length over the computational domain. The correlation length in the periodic hill case is approximated as the height of the hill crest, i.e. $l_c/H = 1$, and without loss of generality, we choose the velocity along profiles at $x/H = 1, 3, 5$ and 7 as the training data. The comparisons of $\theta_1$ and $\theta_2$ between the ensemble-based learned model and the truth are presented in figures 9(*b*) and 9(*c*), respectively. The plots of kernel densities in figures 9(*b*) and 9(*c*) indicate that there are relatively small number of cells with input features $\theta_1$ and $\theta_2$ with large magnitudes. Specifically, only 30 % of the cells in the domain have magnitudes of $\theta_1$ and $\theta_2$ larger than 5.0 and 3.8, respectively. This is the probable cause of the deteriorated estimation in the regions with large $\theta$ magnitudes (figure 9*a*).

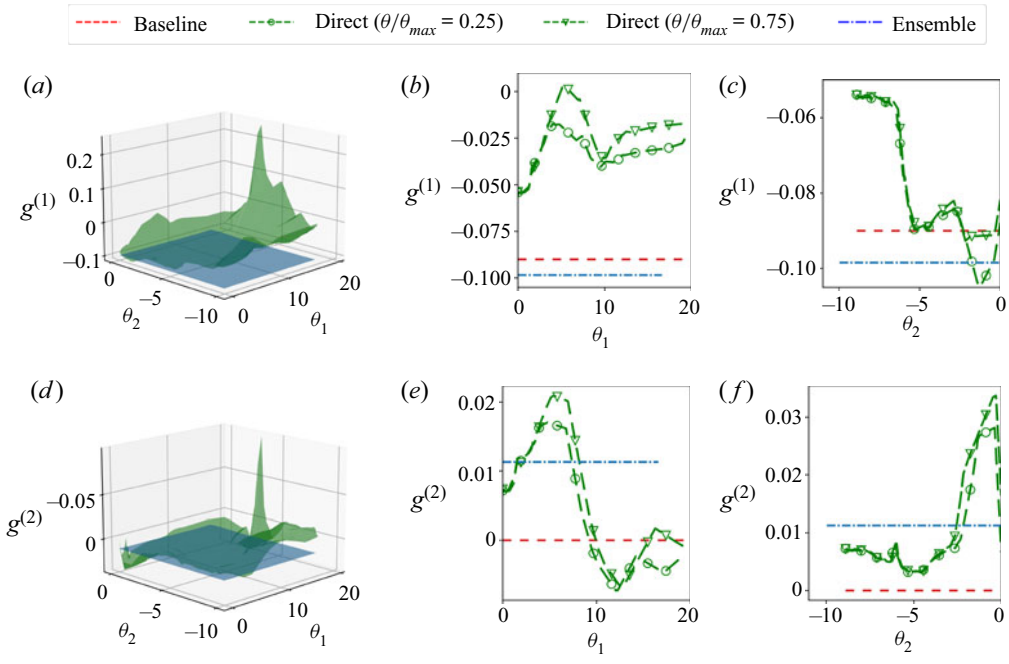



Figure 10. Plots of the mapping between the scalar invariants $\boldsymbol{\theta}$ and the tensor coefficient $\boldsymbol{g}$, with comparison among the baseline model, the direct learned model and the ensemble-based learned model for the periodic hill case. (*a*,*d*) Model functions of $g^{(1)}$ and $g^{(2)}$, respectively, where the blue surface represents the ensemble-based learned model, and the green surface represents the direct learned model. (*b*,*c*,*e*,*f*) Curve plots of the model function at specific planes. For the direct learned model, the plots indicate the learned function at $\theta/\theta_{max} = 0.25$ and 0.75. For the baseline and the ensemble-based learned model, the plots show only the learned function at $\theta/\theta_{max} = 0.25$, since they are almost constant in the entire function space.

The nonlinear mapping between the scalar invariant $\boldsymbol{\theta}$ and the $\boldsymbol{g}$ function is learned from the training data. The functional mappings with the direct learning method and the ensemble-based method are shown in figure 10. In this case, no ground truth of the mapping $\boldsymbol{\theta} \mapsto \boldsymbol{g}$ exists for validation. Here, we show the baseline mapping from the linear eddy viscosity, i.e. $g^{(1)} = -0.09$ and $g^{(2)} = 0$. The direct learned function has relatively strong nonlinearity, while the ensemble-based learned function is almost constant at about $-0.098$ for $g^{(1)}$, and 0.01 for $g^{(2)}$. The $\boldsymbol{g}$ function varies slightly for the large invariant $\theta_1$ and the small invariant $\theta_2$, mainly in the uphill region with large strain rates.

The estimation of turbulence kinetic energy (TKE) and the deviatoric part of Reynolds stress deviates noticeably from the DNS data. The results are presented in figure 11. The direct learned model can estimate the Reynolds stress in relatively better agreement with DNS results than the ensemble-based learned model, due to the use of direct data. As for the turbulence kinetic energy, both the direct learning and ensemble-based methods lead to significant discrepancies compared to the DNS results. The discrepancies can be due to the fact that the divergence-free part of the Reynolds stress has no effect on velocity (Perot 1999), which poses difficulties in reconstructing the Reynolds stress from the velocity data accurately. One can use the velocity field from high-fidelity data to obtain the divergence of the Reynolds stress tensor by balancing the momentum equation and further regard the divergence of the Reynolds stress tensor as the training target to avoid this issue (Cruz *et al.* 2019). Moreover, the large discrepancies in the TKE estimation can be caused by the

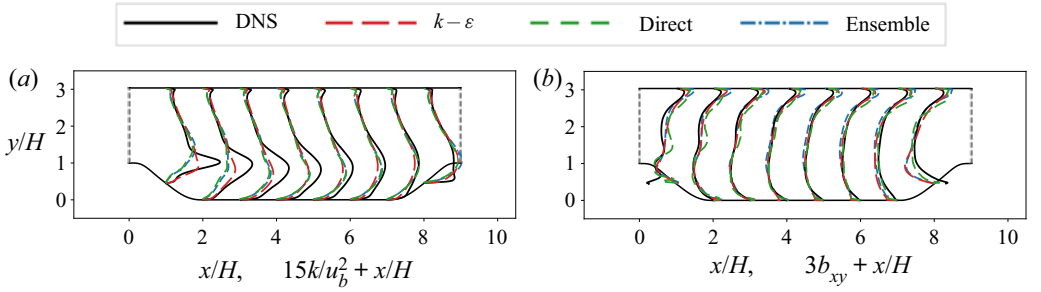



Figure 11. Comparison of (*a*) turbulence kinetic energy, and (*b*) the deviatoric part of Reynolds stress $b_{xy}$, along profiles among the $k$–$\varepsilon$ model, the direct learned model, the ensemble-based learned model and the DNS, for the periodic hill case.

deficiency of the model representation. Specifically, the TKE transport equation can be derived rigorously from the Navier–Stokes equation, and thus it is an exact equation if all terms are modelled correctly. Two terms in the equations are not exact, i.e. the production term $\mathcal{P} = \boldsymbol{\tau} : \boldsymbol{S} = 2k(\boldsymbol{b} + \frac{1}{3}\boldsymbol{I}) : \boldsymbol{S}$, and the dissipation term. The turbulence modelling addresses the modelling of the deviatoric tensor $\boldsymbol{b}$, which if modelled correctly would yield the correct TKE production. However, the dissipation rate is modelled by another transport equation that is much less rigorously derived than the TKE transport equation. The present work focuses on addressing the deficiency of the turbulence closure by learning a nonlinear algebraic Reynolds stress model, i.e. a nonlinear function $\boldsymbol{b} = f(\boldsymbol{S}, \boldsymbol{W})$. This work does not address the shortcomings of the dissipation modelling, which is present in both algebraic models and Reynolds stress transport models (i.e. differential stress models). Nor does it make the stress–strain-strain rate function non-local (e.g. as in Zhou, Han & Xiao 2021; Zhou *et al.* 2022), which is needed for non-equilibrium turbulence requiring Reynolds stress transport models.

Previous efforts of data-driven, Reynolds-stress-based models (Schmelzer *et al.* 2020; Waschkowski *et al.* 2022) have introduced a corrective production term $\delta\mathcal{P}(\boldsymbol{S}, \boldsymbol{W})$ to the TKE transport equation, which improves the TKE prediction. This correction goes beyond the turbulence constitutive modelling and addresses the structure of turbulence quantity transport models. It has the same effects as the multiplicative factor $\beta$ applied to the production term (Singh & Duraisamy 2016). However, note that the latter operates in the realm of the Boussinesq assumption (stress–strain-rate relation), and thus data-driven Reynolds stress models (e.g. Schmelzer *et al.* 2020; Waschkowski *et al.* 2022; and the present work) would introduce more degrees of freedom in the correction if such a production correction term is used.

Our investigation suggests that the nonlinearity of the stress–strain-rate relation may not be the dominant deficiency in the flows studied here. This is evident from figure 11, which shows that the learned model does not significantly improve the TKE estimation. One can also consider $k$ an 'operation variable' described by the TKE transport equation, which is consistent with the widely accepted interpretation that the dissipation rate $\varepsilon$ is an operation variable (Pope 2000). They are not physical variables and do not necessarily need to be compared directly to their DNS counterpart. The purpose of operation variables is to make good predictions of the velocities and their derived field.

Our generalizability test suggests that the learned model with the ensemble-based method is able to generalize to cases that are similar (in terms of feature space) to the trained cases but perform less well in cases with large differences from the trained cases.

| Geometry (slope parameter $\alpha$) | 0.5 | 0.8 | 1.0 | 1.2 | 1.5 |
|---|---|---|---|---|---|
| Maximum of input feature $\theta_1$ | 161 | 115 | 105 | 91 | 69 |
| Exceeds training case ($\alpha = 1$) by | 53.3 % | 9.52 % | 0 | 13.3 % | 34.3 % |

Table 6. Comparison of the maximum values of input features in flow configurations with different slopes $\alpha$.

This test suggests that a wide range of input features should be embedded in order to obtain a practical model. The results of the predicted velocity $u_x$ for different slopes $\alpha$ are shown in figure 12. The prediction with the direct learned model is also presented for comparison. For the case $\alpha = 0.5$, the solver diverges with the direct learned model, hence no results are presented. In the cases $\alpha = 0.8$ and 1.2, the learned model can generally improve the prediction compared to the $k$–$\varepsilon$ model. However, in the case $\alpha = 1.5$, the model leads to significant discrepancies. In contrast, all the cases show that the learned model with the ensemble-based method can noticeably improve the mean flow estimation in terms of the velocity compared to the $k$–$\varepsilon$ model. In particular, for the cases $\alpha = 0.8$ and 1.2, the velocity profiles $u_x$ have a remarkable agreement with the DNS data. That is probably due to the similar input features of these two cases to the training case $\alpha = 1$. Additionally, the error between the prediction and the DNS data over the entire field and the recirculation region ($0 < x/H < 5$ and $0 < y/H < 1$) is shown in figures 12(*e*) and 12(*f*), respectively, where the error from the training case $\alpha = 1$ is also indicated based on the propagated velocity. It is obvious that the ensemble-based learned models provide better prediction than the $k$–$\varepsilon$ model in all the test cases and the direct learned model in all cases except for $\alpha = 0.8$. For the training case ($\alpha = 1$), the learned model provides the lowest prediction error, which is reasonable since the prediction is informed directly by the training data. The model prediction error increases as the extrapolation case is further away from the training case. In particular, noticeable discrepancies are exhibited in the case $\alpha = 1.5$. The maximum value of the input feature is provided in table 6 to show the feature difference among these cases. It can be seen that the range of the input feature for $\alpha = 0.8$ and 1.2 is relatively close to the training case in contrast to the cases $\alpha = 0.5$ and 1.5. This confirms that the consistency of the input features between the training case and the test cases is essential for the generalizability of the data-driven model. For the flow with similar input features, the trained model is able to provide satisfactory predictions. This suggests that a wide range of input features should be included in the training case to obtain a practical model.

## 5. Discussion

### 5.1. *Parallelization*

To enhance the generalizability of the learned model, training data should embed various flow features from different configurations, e.g. the square duct, the periodic hills, and aerofoils. To handle a large data set, the conventional machine learning training algorithms need to split the data randomly into multiple batches. Further, the stochastic gradient descent method is employed to train the model by looping over the entire data set sequentially (Kovachki & Stuart 2019). This makes it inefficient to handle the large data set. The ensemble-based framework is able to learn the model from a large data set in a parallelizable manner. The ensemble-based method is inherently parallelizable and can handle the data with random noise so as to avoid data overfitting. This achieves

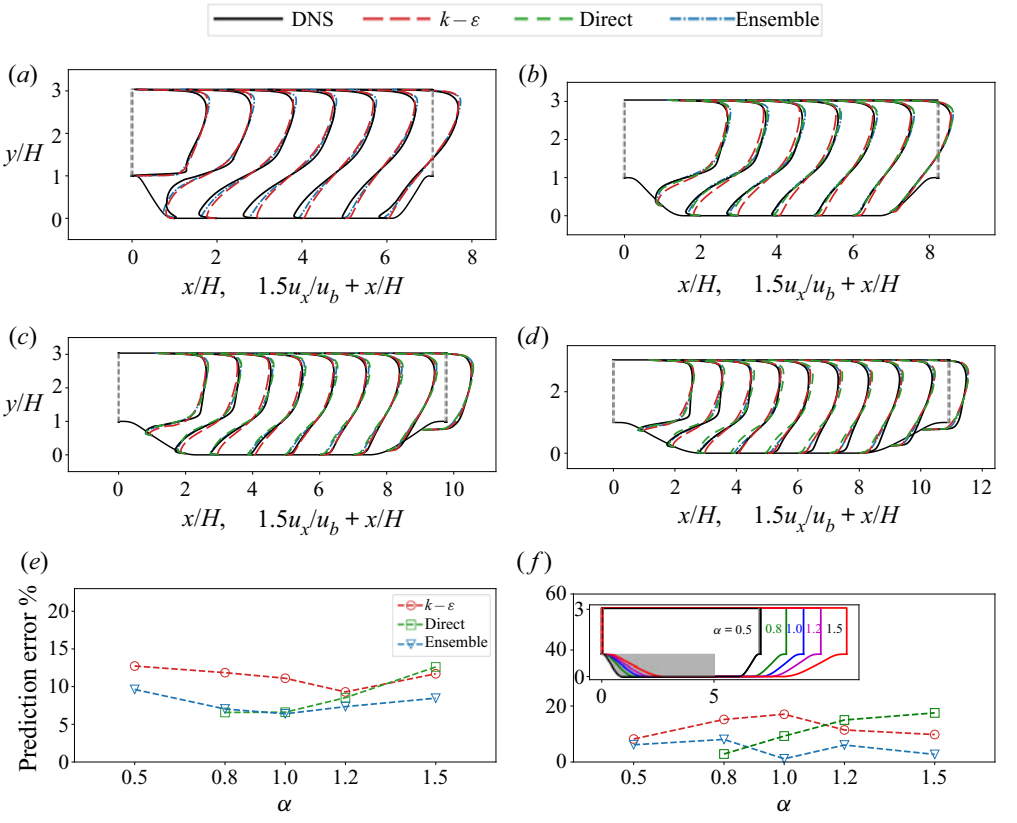



Figure 12. Results of generalizability tests on configurations with different slopes ($\alpha = 0.5, 0.8, 1.2, 1.5$). (*a–d*) Velocity profiles for $\alpha = 0.5, 0.8, 1.2, 1.5$, respectively, with comparison among the $k$–$\varepsilon$ model, the direct learned model, the ensemble-based learned model and the DNS. (*e*,*f*) Plots of the prediction error over the entire field and recirculation region, respectively. The shadow in (*f*) indicates the recirculation region for error calculations. The results of the direct learned model are not shown for the configuration with $\alpha = 0.5$, since the solver diverges in that case.

the same goal as stochastic gradient descent for machine learning. Furthermore, the model-consistent training framework can train with the data from different configurations simultaneously, as noted by Waschkowski *et al.* (2022). These training cases do not need communication (e.g. embarrassingly parallel workload), such that the wall time is not significantly increased when the number of used CPU cores is equal to the number of configurations.

### 5.2. *Flexibility in learning from different observation data*

The ensemble-based framework is extremely flexible in terms of the loss function, specific applications and observation data, due to its derivative-free nature. Specifically, the loss function can even be non-differentiable, e.g. when learning dynamic model parameters with statistical observation data. In such a scenario, the adjoint-based method would be difficult to deploy, while the ensemble-based method needs only to evaluate the cost function and approximate the corresponding gradient based on the model input and output. Moreover, the framework here is used for the turbulence closure problem. Other physical systems where the adjoint solver is not readily available can apply the proposed

method to learn the underlying closure model based on the measurable observations. Also, in specific cases, e.g. RANS modelling, the available data are often collected from different configurations with varying physical quantities and dimensionality. It is difficult for conventional methods to use these disparate data, as they need to develop specific adjoint solvers for different measurable quantities, which is a challenging task for complex CFD solvers. The proposed model-consistent learning framework can approximate the sensitivity of the model prediction to the model parameters based on model inputs and outputs. With its non-intrusive and derivative-free nature, the ensemble-based model-consistent learning is naturally flexible for different loss functions, different physical systems and disparate data.

## 6. Conclusion

This work proposes an ensemble-based framework to learn nonlinear eddy viscosity turbulence models from indirect data. Earlier works (Duraisamy 2021) have noted that there exists an inconsistency issue between training and prediction environments when learning turbulence models from direct data. In this work, we observe that this inconsistency leads to poor generalizability of the learned model. The proposed framework can ensure the consistency and thus improve the generalizability of the learned model. Furthermore, the training method is non-intrusive and does not need an adjoint solver. Moreover, the ensemble-based method has been shown to learn a turbulence model from indirect observation data more efficiently than the adjoint-based method based on our particular implementation of adjoint solver and test cases (both of which are steady-state flows).

The capability of the proposed framework is demonstrated on two flows, the flow in a square duct and the flow over periodic hills. The duct flow demonstrated the capability of the proposed method in learning underlying closure relationships from velocity observation data, and the periodic hill case showed the generalizability of the learned model to flows in similar configurations with varying slopes. Both cases highlight the straightforward implementation of the ensemble-based learning method. It runs in parallel and can learn from large sets of training flows simultaneously. Moreover, the non-intrusive nature of the ensemble-based method makes it convenient to handle different types of observations without developing an adjoint solver for each new objective function.

The main limitations and perspectives of the present ensemble-based learning method are discussed below. First, it would be difficult to apply the present algorithm for scenarios having large data sets, such as unsteady three-dimensional flow data, due to the prohibitive computational cost of large matrix inversion in the update scheme. Dimension reduction techniques such as the truncated singular value decomposition would be incorporated in the ensemble-based method to address this issue (Luo *et al.* 2015). Second, the position of observation data is critical to the training performance. The strategy to select the optimal position of these training data needs to be further investigated. Third, the present framework is based on the nonlinear eddy viscosity model, which is under the weak equilibrium assumption. It would be worthy of investigation to use a neural network to emulate the Reynolds stress transport equation (Zhou *et al.* 2022) with embedded non-equilibrium effects. Also, future works will focus on training with different classes of flows to enhance the generalizability of the learned model, which is a step towards representing a universal or unified turbulence model.

**Acknowledgements.** Figure 1 in this paper was prepared with the help of Mr X. Zhou, whose efforts are greatly appreciated by the authors. Finally, the authors thank the reviewers for their constructive and valuable comments, which greatly improved the quality and clarity of this paper.

**Funding.** X.L.Z. and G.H. are supported by the NSFC Basic Science Center Program for 'Multiscale Problems in Nonlinear Mechanics' (no. 11988102). X.L.Z. also acknowledges support from the National Natural Science Foundation of China (no. 12102435) and the China Postdoctoral Science Foundation (no. 2021M690154). X.L. acknowledges partial financial support from the National Centre for Sustainable Subsurface Utilization of the Norwegian Continental Shelf (NCS2030), Norway. H.X. is not funded when performing this work.

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

**Author ORCIDs.**
Xin-Lei Zhang https://orcid.org/0000-0003-2281-3363;
Heng Xiao https://orcid.org/0000-0002-3323-4028;
Xiaodong Luo https://orcid.org/0000-0002-0734-862X;
Guowei He https://orcid.org/0000-0003-4738-0816.

## Appendix A. Practical implementation

The practical implementation of the proposed ensemble-based model-consistent turbulence modelling framework is detailed in this appendix and illustrated schematically in figure 13. Given the observation error $R$, the data set $y$, and the sample variance $\sigma$, the procedure for the ensemble-based model learning is summarized below.

(i) Pre-training. To obtain the initial weight $\boldsymbol{w}^0$ of the neural network, we pre-train the network to be an equivalent linear eddy viscosity model such that $g^{(1)} = -0.09$ and $g^{(i)} = 0$ (for $i = 2$–$10$). The weights so obtained, $\boldsymbol{w}^0$, are set as the initial value for optimization (Michelén-Ströfer & Xiao 2021). The pre-training is necessary because conventional initialization methods (e.g. random initialization) may lead to non-physical values such as the positive $g_1$ (negative eddy viscosity), which would cause divergence of the RANS solver. Pre-training is needed to address this difficulty and accelerate model learning.

(ii) Initial sampling. We assume that the weights are independent and identically distributed Gaussian random variables with mean $\boldsymbol{w}^0$ and variance $\sigma^2$. We draw random samples of the weights (figure 13*a*) via the formula $\boldsymbol{w}_j = \boldsymbol{w}^0 + \boldsymbol{\epsilon}_j$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$.

(iii) Feature extraction. The velocity field $\boldsymbol{u}$ and turbulence time scale $k/\varepsilon$ are used to compute the scalar invariants $\boldsymbol{\theta}$ and the tensor bases $\boldsymbol{T}$ (figure 13*b*) based on (2.5) and (2.6*a*,*b*). The scalar invariants are then adopted as the inputs of the neural network function $\boldsymbol{g}$, while the tensor bases are employed to construct the Reynolds stress by combining with the outputs of the neural network as illustrated in step (iv) below. The input features of the neural network are scaled into the range [0, 1] with the maximum and minimum values. In posterior tests, the maximum and minimum values in the training case need to be used to scale the input features in test cases. Note that this strategy uses global quantities, i.e. the maximum and minimum values of the scalar invariants, to normalize the input features. Such global normalization may lead to feature clustering along certain directions within the feature space, and cause further convergence issues. In particular, when training jointly with different classes of flows, the input features can have a wide diversity of the scalar invariants $\boldsymbol{\theta}$.

For example, the flow with shock waves can provide significant extreme values of scalar invariants. Using these values to scale the input features of other flows would lead to feature clustering in a narrow range near 0 in posterior tests or joint training. In such scenarios, it is necessary to normalize the input features based on local quantities (Ling & Templeton 2015; Wang *et al.* 2017; Wu, Xiao & Paterson 2018), e.g. $\hat{\theta} = \theta/(|\theta| + |\theta^*|)$, where $\theta^*$ is local normalization. Such a normalization can ensure that the normalized quantity $\hat{\theta}$ falls within the range $[-1, 1]$. The local normalization can avoid the feature clustering issue with appropriate choices of the normalization factor $\theta^*$, which is worthy of further investigation in future studies.

(iv) Evaluation of Reynolds stress. Input features $\boldsymbol{\theta}$ are propagated to the basis coefficient $\boldsymbol{g}$ with each realization of the weights $\boldsymbol{w}$, and then the Reynolds stress can be constructed (figure 13*c*) through combining the coefficient $\boldsymbol{g}$ and the tensor basis $\boldsymbol{T}$, i.e. $\boldsymbol{\tau} = 2k\sum_i g^{(i)}\boldsymbol{T}^{(i)} + \frac{2}{3}k\boldsymbol{I}$.

(v) Propagation to velocity. The velocity is obtained by solving the RANS equations for each constructed Reynolds stress. Moreover, the turbulence kinetic energy and the dissipation rate are obtained by solving the turbulence transport equations (figure 13*d*).

(vi) Computation of Kalman gain from samples. To this end, we first compute the square-root matrices at iteration step $l$ as follows:

$$S_w^l = \frac{1}{\sqrt{N_e - 1}}\left[\boldsymbol{w}_1^l - \bar{\boldsymbol{w}}^l, \boldsymbol{w}_2^l - \bar{\boldsymbol{w}}^l, \ldots, \boldsymbol{w}_{N_e}^l - \bar{\boldsymbol{w}}^l\right], \tag{A1a}$$

$$S_y^l = \frac{1}{\sqrt{N_e - 1}}\left[\mathcal{H}[\boldsymbol{w}_1^l] - \mathcal{H}[\bar{\boldsymbol{w}}^l], \mathcal{H}[\boldsymbol{w}_2^l] - \mathcal{H}[\bar{\boldsymbol{w}}^l], \ldots, \mathcal{H}[\boldsymbol{w}_{N_e}^l] - \mathcal{H}[\bar{\boldsymbol{w}}^l]\right], \tag{A1b}$$

$$\bar{\boldsymbol{w}}^l = \frac{1}{N_e}\sum_{j=1}^{N_e}\boldsymbol{w}_j^l, \tag{A1c}$$

where $N_e$ is the sample size. The Kalman gain matrix is then computed as

$$K = S_w S_y^{\mathrm{T}}\left(S_y S_y^{\mathrm{T}} + \gamma^l R\right)^{-1}. \tag{A2}$$

(vii) Update weights of neural networks. Use the iterative ensemble Kalman method to update the weights of the neural network (figure 13*e*), i.e.

$$\boldsymbol{w}_j^{l+1} = \boldsymbol{w}_j^l + K\left(y_j - \mathcal{H}[\boldsymbol{w}_j^l]\right). \tag{A3}$$

In steps (vi) and (vii), the parameter $\gamma$ is adjusted in an inner loop. This inner loop adjusts adaptively the update step length by inflating the observation error covariance with the parameter $\gamma$. Specifically, we let $\gamma^\upsilon = \beta^\upsilon\{S_y^\upsilon(S_y^\upsilon)^{\mathrm{T}}\}/\{R\}$, where $\beta^\upsilon$ is a scalar coefficient whose value changes at each sub-iteration index $\upsilon$. Specifically, at each iteration, an initial value (i.e. at sub-iteration step $\upsilon = 0$) is set to be $\beta^0 = 1$. If at the $\upsilon$th sub-iteration step, the average data misfit (over the ensemble of model predictions) is reduced, then at the next sub-iteration step, we set $\beta^{\upsilon+1} = 0.8\beta^\upsilon$ and break out of the inner loop; otherwise, we set $\beta^{\upsilon+1} = 1.2\beta^\upsilon$, and repeat step (vi). We allow up to five sub-iterations in this inner loop.

(viii) If the ensemble variance is smaller than the observation error, then consider the iteration converged and end the iteration; otherwise, continue to step (iii) until the convergence criterion above is met or the maximum number of iterations is reached.

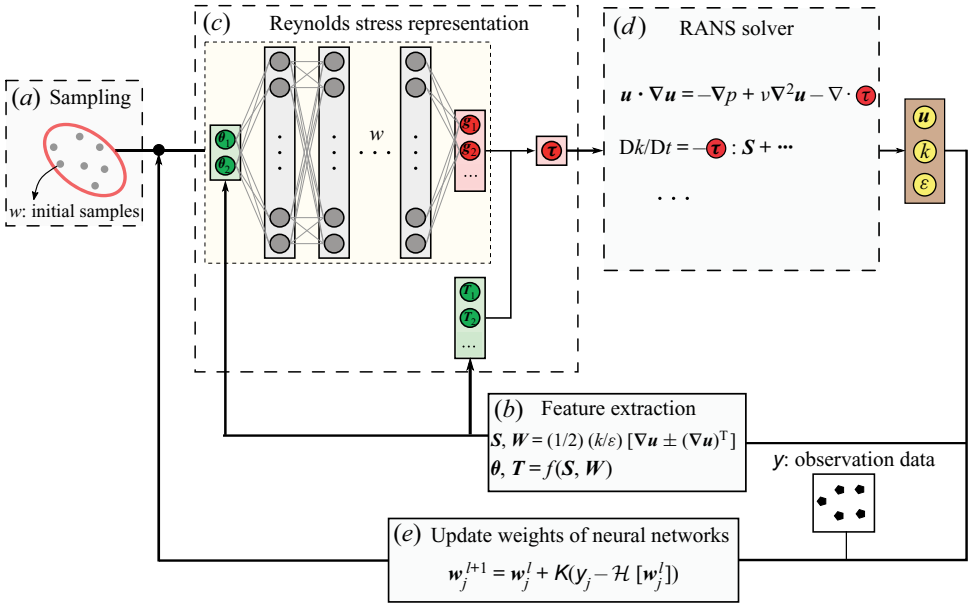



Figure 13. Detailed schematic of ensemble-based model-consistent training of the tensor basis neural network: (*a*) generate samples of neural network weights; (*b*) extract input features; (*c*) evaluate Reynolds stress based on tensor basis neural network; (*d*) propagate the Reynolds stress to velocities; (*e*) update weights of neural networks based on ensemble Kalman method.

## Appendix B. Hessian matrix in the ensemble Kalman method

In this appendix, we illustrate how the approximated Hessian matrix, as well as the gradient (Jacobian), are incorporated implicitly in the ensemble Kalman method, which leads to accelerated learning and improved robustness. This is a crucial difference compared to the stochastic gradient descent optimization used for neural network training in deep learning.

The weight-update scheme of the iterative ensemble Kalman method is formulated as in (2.9):

$$\boldsymbol{w}_j^{l+1} = \boldsymbol{w}_j^l + K\left(y_j - \mathcal{H}[\boldsymbol{w}_j^l]\right) \quad \text{with } K = S_w S_y^{\mathrm{T}}\left(S_y S_y^{\mathrm{T}} + \gamma^l R\right)^{-1}. \tag{2.9}$$

We first establish its connection to the common form of the Kalman gain matrix $K = PH^{\mathrm{T}}(HPH^{\mathrm{T}} + \gamma^l R)^{-1}$. To this end, we write the model error covariance matrix $P$ and other associated quantities in terms of the square-root matrix $S_w$ and its projection $S_y$ to the observation space, i.e.

$$P = S_w S_w^{\mathrm{T}} \quad \text{and} \quad S_y = HS_w. \tag{B1$a$,$b$}$$

Consequently, the cross-covariance $PH^{\mathrm{T}}$ between the weights $\boldsymbol{w}$ and the predictions $\mathcal{H}[\boldsymbol{w}]$, and the projection of $P$ to the observation space, are

$$PH^{\mathrm{T}} = S_w S_y^{\mathrm{T}} \quad \text{and} \quad HPH^{\mathrm{T}} = S_y S_y^{\mathrm{T}}, \tag{B2$a$,$b$}$$

respectively. The two forms of the Kalman gain matrix are thus established.

Next, we show that the Kalman gain matrix $K$ in the update scheme implicitly contains the inverse of an approximated Hessian matrix of the cost function. To see this, let $H$ be

the local gradient of the observation operator $\mathcal{H}$ (with respect to the parameter $\boldsymbol{w}$ – and the same for all gradients and Hessians mentioned hereafter). After dropping the iteration index, it can be shown that the gradient of the data misfit term $J' = \|y - \mathcal{H}[\boldsymbol{w}]\|^2_{\gamma R}$ in (2.8) is given by (neglecting a constant factor 2)

$$\frac{\partial J'}{\partial \boldsymbol{w}} = -H^{\mathrm{T}}(\gamma R)^{-1}\left(y_j - \mathcal{H}[\boldsymbol{w}_j]\right), \tag{B3}$$

and the local Hessian matrix of the entire objective function is given by (neglecting a constant factor 2)

$$\frac{\partial^2 J}{\partial \boldsymbol{w}^2} = P^{-1} + H^{\mathrm{T}}(\gamma R)^{-1}H. \tag{B4}$$

We will utilize the matrix identity

$$PH^{\mathrm{T}}\left(HPH^{\mathrm{T}} + \gamma R\right)^{-1} = \left(P^{-1} + H^{\mathrm{T}}(\gamma R)^{-1}H\right)^{-1} H^{\mathrm{T}}(\gamma R)^{-1}; \tag{B5}$$

see (49) in Luo (2021) for detailed derivations of this identity. In general, the observation operator $\mathcal{H}$ is nonlinear, in which case the square-root matrix $S_y$ as estimated in (A1) provides a derivative-free approximation to the projected square-root matrix $HS_w$. Accordingly, one can see that the term $K(y_j - \mathcal{H}[\boldsymbol{w}_j^l])$ in (2.9) is an ensemble-based derivative-free approximation to the product of the inverse of the local Hessian matrix in (B4) and the (negative) local gradient in (B3). In other words, the weight-update formula (2.9) implicitly utilizes the information of both approximated gradient and Hessian matrices.

## Appendix C. Sensitivity study of network architecture and observation data

Neural networks with different architectures are used in the model-consistent training of the square duct case to show the sensitivity of the framework. Three network architectures are tested: (1) two hidden layers with 5 neurons per layer (baseline); (2) two hidden layers with 10 neurons per layer; and (3) ten hidden layers with 10 neurons per layer. The results of errors in the velocity and Reynolds stress over the entire field are summarized in table 7. It can be seen that the results are not very sensitive to the neural network architecture for the square duct case. The errors in the velocity and the Reynolds stress among the three cases are very similar. It is noted that the case with two layers and five neurons per layer is able to predict well the flow fields in both velocities and Reynolds stresses. This is likely due to the narrow range of the input features in this case. The maximum of the input features is approximately 7, which can be captured sufficiently with 69 parameters in the neural network. Moreover, we test the setting of using the velocity observation along the anti-diagonal line of the computational domain. The results in both the velocity and the Reynolds stress field become slightly inferior compared to the case with the full field.

We perform the sensitivity analysis on the neural network for the periodic hill case. The neural network with ten hidden layers and 10 neurons per layer is regarded as the baseline since it has been used in the work of Ling *et al.* (2016). Moreover, we test the neural network with two hidden layers and five neurons per layer, and the neural network with two hidden layers and 10 neurons per layer. The results are summarized in table 8. In general, the velocity and Reynolds stress results do not vary significantly with different neural networks. In other words, the training performance is not sensitive to the neural

| Network architecture (neurons/layer × layers) | 5 × 2 | 10 × 2 | 10 × 10 | 5 × 2 (fewer data) |
|---|---|---|---|---|
| Number of weights | 69 | 184 | 1064 | 69 |
| Number of data points | 2500 | 2500 | 2500 | 50 |
| Error in mean velocities, $\mathcal{E}(\boldsymbol{u})$ | 0.47 % | 0.91 % | 0.52 % | 2.0 % |
| Error in Reynolds stresses, $\mathcal{E}(\boldsymbol{\tau})$ | 5.8 % | 6.9 % | 6.0 % | 9.4 % |

Table 7. Sensitivity of predictive performance to network architecture and observation data for the square duct case.

| Network architecture (neurons/layer × layers) | 5 × 2 | 10 × 2 | 10 × 10 |
|---|---|---|---|
| Number of weights | 57 | 162 | 1042 |
| Error in mean velocities, $\mathcal{E}(\boldsymbol{u})$ | 6.5 % | 6.6 % | 6.4 % |
| Error in Reynolds stresses, $\mathcal{E}(\boldsymbol{\tau})$ | 43.0 % | 45.0 % | 44.0 % |

Table 8. Sensitivity of predictive performance to network architecture for the periodic hill case.

network architecture. Hence we choose the baseline network with ten hidden layers and 10 neurons per layer in this case.

In summary, the neural network architecture has no significant effects on the training accuracy based on the sensitivity study for both the square duct and periodic hill cases. Hence we choose the baseline networks for both cases. That is, the network with two hidden layers and five neurons per layer is used for the square duct case, and the network with ten hidden layers and 10 neurons per layer is adopted for the periodic hill case. Additionally, this choice keeps consistency to the previous works (Michelén-Ströfer & Xiao 2021; Michelén-Ströfer *et al.* 2021*b*) such that the training performances with different approaches shown in table 2 are compared in a consistent manner.

## Appendix D. Implementation of the direct learning method

This appendix presents briefly the implementation of the direct learning method (Ling *et al.* 2016). The deviatoric part of the Reynolds stress tensor, $\boldsymbol{b}$, from DNS is used as the training data to optimize the tensor basis neural network. The input features of scalar invariants are from the baseline RANS prediction. The cost function can be written as

$$J = \|\boldsymbol{g}[\boldsymbol{w}]\,\boldsymbol{T} - \boldsymbol{b}^{DNS}\|^2 + \lambda\,\|\boldsymbol{g}[\boldsymbol{w}] - \boldsymbol{g}_0\|^2, \tag{D1}$$

where the tensor bases $\boldsymbol{T}$ are obtained from the baseline RANS results, and $\boldsymbol{g}_0$ represents the prior value, i.e. $g_0^{(1)} = -0.09$ and $g_0^{(2)-(10)} = 0$. The model training amounts to finding optimal weights of the neural network such that the cost function $J$ is minimized. In the region where the tensor basis $\boldsymbol{T}$ is small, the coefficients $\boldsymbol{g}$ can have extremely large, non-physical values, which leads to negative eddy viscosity. Hence the regularization is required to alleviate the ill-conditioning by penalizing the deviation from the prior value. The regularization parameter $\lambda$ is adjusted to achieve good data fit and avoid extreme values of $\boldsymbol{g}$ simultaneously. It is taken as 10 in this work. To solve the least-squares

problem, the gradient of the cost function is obtained based on auto-differentiation, and the Adam algorithm is adapted to update the weights of the neural network.

## References


Abadi, M., *et al.*2015 TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bae, H.J. & Koumoutsakos, P. 2022 Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nat. Commun.* **13** (1), 1443.

Brener, B.P., Cruz, M.A., Thompson, R.L. & Anjos, R.P. 2021 Conditioning and accurate solutions of Reynolds average Navier–Stokes equations with data-driven turbulence closures. *J. Fluid Mech.* **915**, A110.

Chen, Y., Chang, H., Meng, J. & Zhang, D. 2019 Ensemble neural networks (ENN): a gradient-free stochastic method. *Neural Netw.* **110**, 170–185.

Chen, Y. & Oliver, D.S. 2013 Levenberg–Marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification. *Comput. Geosci.* **17** (4), 689–703.

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

Duraisamy, K. 2021 Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. *Phys. Rev. Fluids* **6** (5), 050504.

Eisfeld, B., Rumsey, C. & Togiti, V. 2016 Verification and validation of a second-moment-closure model. *AIAA J.* **54** (5), 1524–1541.

Evensen, G. 2009 *Data Assimilation: the Ensemble Kalman Filter*. Springer.

Evensen, G. 2018 Analysis of iterative ensemble smoothers for solving inverse problems. *Comput. Geosci.* **22** (3), 885–908.

Han, J., Zhou, X.-H. & Xiao, H. 2022 VCNN-e: a vector-cloud neural network with equivariance for emulating Reynolds stress transport equations. arXiv:2201.01287.

Holland, J.R., Baeder, J.D. & Duraisamy, K. 2019 Field inversion and machine learning with embedded neural networks: physics-consistent neural network training. *AIAA Aviation 2019 Forum*. *AIAA Paper* 2019-3200.

Kovachki, N.B. & Stuart, A.M. 2019 Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Probl.* **35** (9), 095005.

Launder, B.E., Reece, G.J. & Rodi, W. 1975 Progress in the development of a Reynolds-stress turbulence closure. *J. Fluid Mech.* **68** (3), 537–566.

Launder, B.E. & Sandham, N.D. 2002 *Closure Strategies for Turbulent and Transitional Flows*. Cambridge University Press.

Launder, B.E. & Sharma, B.I. 1974 Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Lett. Heat Mass Transfer* **1** (2), 131–137.

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

Ling, J. & Templeton, J. 2015 Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. *Phys. Fluids* **27** (8), 085103.

Luo, X. 2021 Novel iterative ensemble smoothers derived from a class of generalized cost functions. *Comput. Geosci.* **25** (3), 1159–1189.

Luo, X., Bhakta, T. & Naevdal, G. 2018 Correlation-based adaptive localization with applications to ensemble-based 4D-seismic history matching. *SPE J.* **23** (02), 396–427.

Luo, X., Stordal, A.S., Lorentzen, R.J. & Nævdal, G. 2015 Iterative ensemble smoother as an approximate solution to a regularized minimum-average-cost problem: theory and applications. *SPE J.* **20** (05), 962–982.

MacArt, J.F., Sirignano, J. & Freund, J.B. 2021 Embedded training of neural-network subgrid-scale turbulence models. *Phys. Rev. Fluids* **6** (5), 050502.

Michelén-Ströfer, C.A. & Xiao, H. 2021 End-to-end differentiable learning of turbulence models from indirect observations. *Theor. Appl. Mech. Lett.* **11** (4), 100280.

Michelén-Ströfer, C.A., Zhang, X.-L. & Xiao, H. 2021*a* DAFI: an open-source framework for ensemble-based data assimilation and field inversion. *Commun. Comput. Phys.* **29** (5), 1583–1622.

Michelén-Ströfer, C.A., Zhang, X.-L. & Xiao, H. 2021*b* Ensemble gradient for learning turbulence models from indirect observations. *Commun. Comput. Phys.* **30** (5), 1269–1289.

Nocedal, J. & Wright, S. 2006 *Numerical Optimization*. Springer.

NOVATI, G., DE LAROUSSILHE, H.L. & KOUMOUTSAKOS, P. 2021 Automating turbulence modelling by multi-agent reinforcement learning. *Nat. Mach. Intell.* **3** (1), 87–96.

OTHMER, C. 2008 A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *Intl J. Numer. Meth. Fluids* **58** (8), 861–877.

PARK, J. & CHOI, H. 2021 Toward neural-network-based large eddy simulation: application to turbulent channel flow. *J. Fluid Mech.* **914**, A16.

PEROT, B. 1999 Turbulence modeling using body force potentials. *Phys. Fluids* **11** (9), 2645–2656.

POPE, S.B. 1975 A more general effective-viscosity hypothesis. *J. Fluid Mech.* **72** (2), 331–340.

POPE, S.B. 2000 *Turbulent Flows*. Cambridge University Press.

SAÏDI, I.B.H., SCHMELZER, M., CINNELLA, P. & GRASSO, F. 2022 CFD-driven symbolic identification of algebraic Reynolds-stress models. *J. Comput. Phys.* **457**, 111037.

SCHMELZER, M., DWIGHT, R.P. & CINNELLA, P. 2020 Discovery of algebraic Reynolds-stress models using sparse symbolic regression. *Flow Turbul. Combust.* **104** (2), 579–603.

SCHNEIDER, T., STUART, A.M. & WU, J.-L. 2020*a* Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *J. Comput. Phys.* **470**, 111559.

SCHNEIDER, T., STUART, A.M. & WU, J.-L. 2020*b* Imposing sparsity within ensemble Kalman inversion. arXiv:2007.06175v1.

SHIH, T.-H. 1993 A realizable Reynolds stress algebraic equation model. NASA-TM-105993. National Aeronautics and Space Administration.

SINGH, A.P. & DURAISAMY, K. 2016 Using field inversion to quantify functional errors in turbulence closures. *Phys. Fluids* **28** (4), 045110.

SIRIGNANO, J. & SPILIOPOULOS, K. 2022 Online adjoint methods for optimization of PDEs. *Appl. Maths Optim.* **85**, 18.

SLOTNICK, J., KHODADOUST, A., ALONSO, J., DARMOFAL, D., GROPP, W., LURIE, E. & MAVRIPLIS, D. 2014 CFD vision 2030 study: a path to revolutionary computational aerosciences. NASA Tech. Rep. CR-2014-218178.

SPALART, P.R. 2000 Strategies for turbulence modelling and simulations. *Intl J. Heat Fluid Flow* **21** (3), 252–263.

SPALART, P.R. & ALLMARAS, S.R. 1992 A one-equation turbulence model for aerodynamic flows. *AIAA Paper* 1992-439.

SPEZIALE, C.G., SARKAR, S. & GATSKI, T.B. 1991 Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach. *J. Fluid Mech.* **227**, 245–272.

SUN, L. & WANG, J.-X. 2020 Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theor. Appl. Mech. Lett.* **10** (3), 161–169.

THE OPENFOAM FOUNDATION 2021 *OpenFOAM User Guide*.

WALLIN, S. & JOHANSSON, A.V. 2000 An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *J. Fluid Mech.* **403**, 89–132.

WANG, J.-X., WU, J.-L. & XIAO, H. 2017 Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Phys. Rev. Fluids* **2** (3), 034603.

WASCHKOWSKI, F., ZHAO, Y., SANDBERG, R. & KLEWICKI, J. 2022 Multi-objective CFD-driven development of coupled turbulence closure models. *J. Comput. Phys.* **452**, 110922.

WEATHERITT, J. & SANDBERG, R. 2016 A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. *J. Comput. Phys.* **325**, 22–37.

WU, J.-L., MICHELÉN-STRÖFER, C.A. & XIAO, H. 2019*a* Physics-informed covariance kernel for model-form uncertainty quantification with application to turbulent flows. *Comput. Fluids* **193**, 104292.

WU, J.-L., XIAO, H. & PATERSON, E. 2018 Physics-informed machine learning approach for augmenting turbulence models: a comprehensive framework. *Phys. Rev. Fluids* **3** (7), 074602.

WU, J.-L., XIAO, H., SUN, R. & WANG, Q. 2019*b* Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned. *J. Fluid Mech.* **869**, 553–586.

XIAO, H. & CINNELLA, P. 2019 Quantification of model uncertainty in RANS simulations: a review. *Prog. Aerosp. Sci.* **108**, 1–31.

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

YANG, X.I.A. & GRIFFIN, K.P. 2021 Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation. *Phys. Fluids* **33** (1), 015108.

ZHANG, X.-L., MICHELÉN-STRÖFER, C.A. & XIAO, H. 2020*a* Regularized ensemble Kalman methods for inverse problems. *J. Comput. Phys.* **416**, 109517.

ZHANG, X.-L., XIAO, H., GOMEZ, T. & COUTIER-DELGOSHA, O. 2020*b* Evaluation of ensemble methods for quantifying uncertainties in steady-state CFD applications with small ensemble sizes. *Comput. Fluids* **203**, 104530.

ZHANG, X.-L., XIAO, H., LUO, X. & HE, G. 2022 Ensemble-based learning of turbulence models. Software available from github.com/xiaoh/DAFI/ensemble-learning.

ZHAO, Y., AKOLEKAR, H.D., WEATHERITT, J., MICHELASSI, V. & SANDBERG, R.D. 2020 RANS turbulence model development using CFD-driven machine learning. *J. Comput. Phys.* **411**, 109413.

ZHOU, X.-H., HAN, J. & XIAO, H. 2021 Learning nonlocal constitutive models with neural networks. *Comput. Meth. Appl. Mech. Engng* **384**, 113927.

ZHOU, X.-H., HAN, J. & XIAO, H. 2022 Frame-independent vector-cloud neural network for nonlocal constitutive modeling on arbitrary grids. *Comput. Meth. Appl. Mech. Engng* **388**, 114211.