Using delayed detached Eddy simulation to create datasets for data-driven turbulence modeling: A periodic hills with parameterized geometry case

Computers and Fluids 288 (2025) 106506

Contents lists available at ScienceDirect

Computers and Fluids

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

Using Delayed Detached Eddy Simulation to create datasets for data-driven turbulence modeling: A periodic hills with parameterized geometry case

Davide Oberto [a,*], Davide Fransos [b], Stefano Berrone [a]

[a] Politecnico di Torino, Department of Mathematical Sciences, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
[b] Sauber Motorsport AG, Wildbachstrasse 9, 8340 Hinwil, Switzerland

ARTICLE INFO

Keywords:
Turbulence modeling
Neural networks
RANS closure
DDES turbulence models
Periodic hills

ABSTRACT

Despite the emerging field of data-driven turbulence models, there is a lack of systematic high-fidelity datasets at flow configurations changing continuously with respect to geometrical/physical parameters. In this work, we investigate the possibility of using Delayed Detached Eddy Simulation (DDES) to generate reliable datasets in a significantly cheaper manner compared to the DNS or LES counterparts. To do that, we perform 25 simulations of the geometrically-parameterized periodic hills test case to deal with different hills steepnesses. We firstly check the accuracy of our results by comparing one simulation with the benchmark case of Xiao et al. Then, we use such database to train the turbulent viscosity-Vector Basis Neural Network (ν_t-VBNN) data-driven turbulence model. The latter outperforms the classic $k-\omega$ SST RANS model, proving that our generated dataset can be useful for data-driven turbulence modeling and opening the opportunity to exploit DDES to create systematic datasets for data-driven turbulence modeling.

1. Introduction

Albeit the majority of industrial and daily-life flows are turbulent, we are still lacking to fully understand and accurately model turbulence. This aspect drastically impacts on Computational Fluid Dynamics (CFD): resolving Direct Numerical Simulation (DNS) is computational prohibitive and, thus, the usage of turbulence models is almost mandatory.

Turbulence models can be divided into two main groups: (i) scale-resolving models, such as Large Eddy Simulation (LES), where the larger scales of turbulence are resolved and only the small ones are modeled; (ii) completely modeling models, such as Reynolds-Averaged Navier–Stokes (RANS) equations where all turbulence contribution on the averaged fields is modeled. The former are usually more accurate but still require considerable computational cost, making them unaffordable for many industrial cases [1]. Consequently, RANS equations are the most common choice to tackle highly turbulent flows with complicated geometries.

Despite the long history of the RANS modeling field [2], RANS models are currently experiencing a stagnation period in terms of accuracy improvements [3]. This consideration, combined with the increasing interest in Scientific Machine Learning, has pushed researches to use machine learning techniques in the RANS field [4–7].

While the common aim is to improve the accuracy of data-driven RANS models compared to the classic ones, authors have investigated different aspects of this emerging field. For example, several works focus on enforcing into data-driven models physical constraints such as Galilean invariance or independence of the frame-reference coordinates [8–10] or on dealing with the bad-conditioning of the resulting RANS system [11,12]. Others analyze the target turbulent quantities to be predicted through machine learning [13–16] or the machine learning framework to be used [17–22].

As pointed out by Xiao et al. [23], while more and more LES and DNS data are becoming available for various flow typologies, the field still lacks of systematic datasets with fixed flow configuration and parameterized by geometrical and/or physical parameters. Honorable mention must be made to the Pinelli et al. [24] dataset of DNS results of flows in a square duct at different Reynolds numbers. In an attempt to fill the gap, Xiao et al. created a database by performing several DNS simulations of the well-known and investigated Periodic Hill (PH) case [25]. At the begging, this dataset contained data coming from 5 DNS simulations with different hill's steepnesses and, since then, the number of simulations available has increased. However, adding new data coming from DNS simulations is computationally expensive and makes prohibitive the generation of datasets spanning wide parametric spaces, tackling complicate geometries or at high Reynolds numbers.

For this reason, in this paper we investigate the possibility to use Delayed Detached Eddy Simulation (DDES) to generate such datasets.

* Corresponding author.
E-mail address: davide.oberto@polito.it (D. Oberto).

https://doi.org/10.1016/j.compfluid.2024.106506
Received 30 July 2024; Received in revised form 15 October 2024; Accepted 27 November 2024
Available online 4 December 2024

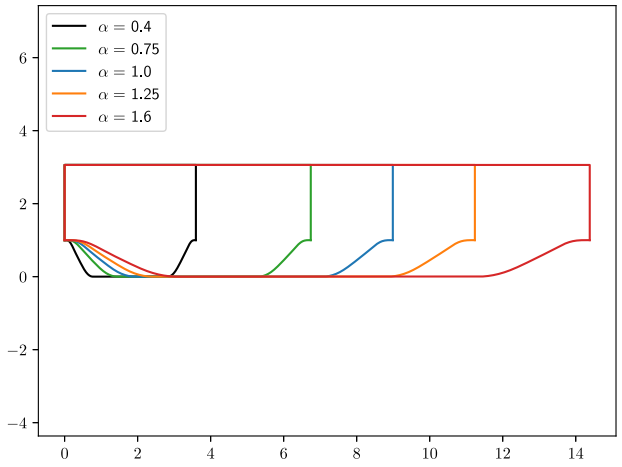


Fig. 1. Parametric hill shapes depending on α. Each geometry is obtained by multiplying the x-coordinates of the reference case by α.

DDES models, specific cases of Detached Eddy Simulation (DES) models, can be interpreted as hybrid RANS-LES models where a RANS model is used in some regions while LES is used in the remaining part of the domain. The switch among models is made by means of a blending function [26]. The scope of using DDES models in this context is to build cheaper but still reliable datasets. To investigate the feasibility of this approach, we create a new dataset composed by 25 simulations of the PH case. The geometries of the simulations are geometrically parameterized by a factor α: each geometry is obtained from the reference one ($\alpha = 1$) by multiplying the x-coordinates of the reference case by α, see Fig. 1. This parameter assumes values $\alpha = 0.4, 0.45, \dots, 1.55, 1.6$. To our best knowledge, no similar one-parameter coverage has been presented in the data-driven turbulence models literature. We observe that, except for the $\alpha = 1$ case, the simulations domains differ from the Xiao et al. ones. Indeed, in [23] the α parameter affects the hills steepness but not the length of the domain between the two hills. On the other hand, in the present study α affects the shape of all the domain. Successively, we exploit such data to train and test the ν_t-VBNN [27] model. This model predicts through two neural networks the turbulent viscosity ν_t and the Reynolds force vector, i.e. the divergence of the Reynolds stress tensor, contribution not accounted in the turbulent viscosity term. The particular architecture and inputs choice of this approach guarantee both Galilean invariance and appropriate rotations under change of coordinates [10]. We show that the ν_t-VBNN model trained with our dataset significantly outperforms the standard $k-\omega$ SST model.

The paper is structured as follows: besides this introduction, in the next Section we make a brief overview on turbulence modeling aspects useful to this work; successively, in Section 3 we describe the numerical framework of our DDES simulations and we carry out a careful analysis using the DNS benchmark results of Xiao et al. to check the accuracy of the $\alpha = 1$ case; then, we discuss in Section 4 the data-driven RANS framework employed in this work; in Section 5 we analyze the results obtained with our data-driven model and study the effect of some training parameters; finally, conclusions are drawn.

As a final remark, we share our DDES results on GitHub [28].

2. Turbulence models

Classic Navier–Stokes (NS) equations for incompressible flows read

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} = -\nabla p, \end{cases} \tag{1}$$

where $\boldsymbol{u}$ is the velocity field, p the pressure field divided by the constant mass density of the fluid and ν is the kinematic viscosity of the fluid. Additionally, we make the assumption that we deal with flows that are steady in-average. While NS equations are perfectly suited to mathematically describe turbulent flows, they require a computationally prohibitive discretization in both time and space to be able to describe the smallest eddies [29,30]. As a matter of fact, turbulence models are usually used to model the effects of all turbulence scales or smallest turbulence scales to mitigate the constraint on the computational discretization.

In the following, we briefly describe three classes of turbulence models: Reynolds-averaged Navier–Stokes (RANS) models, Large Eddy Simulation (LES) and Detached Eddy Simulation (DES).

2.1. RANS models

Steady RANS equations for incompressible flows read

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} = -\nabla p - \nabla \cdot \boldsymbol{\tau}, \end{cases} \tag{2}$$

where, with an abuse of notation, $\boldsymbol{u}$ and p represent the time-averaged velocity and pressure field respectively. $\boldsymbol{\tau}$ is the *Reynolds stress tensor* (RST), a symmetric second order tensor that takes into account the correlation among the fluctuating components of the velocity field. This tensor is unknown and need to be closed. We also define the *Reynolds force vector* $\boldsymbol{t}$ as the divergence of the RST [13].

Classically, RANS models define a set of partial differential equations to close the RST. Here, we focus on linear models based on the Boussinesq's hypothesis

$$\boldsymbol{\tau} = \frac{2}{3} k \boldsymbol{I} - 2\nu_t \boldsymbol{S}, \tag{3}$$

where $k = 1/2\ \mathrm{tr}(\boldsymbol{\tau})$ in the *turbulent kinetic energy*, being tr the trace operator, $\boldsymbol{I}$ is the identity tensor, ν_t is the *turbulent viscosity* and $\boldsymbol{S} = 1/2\ [\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T]$ is the mean strain rate tensor.

In this work we use both the $k-\omega$ SST model [31] and $k-\varepsilon$ model [32]. In the former, we have $\nu_t = k/\omega$, being ω the turbulence specific dissipation rate, while in the latter we have $\nu_t = 0.09k/\varepsilon^2$. Two transport equations for k and either ω or ε are solved.

2.2. LES models

Large eddy simulation (LES) solve the equations

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} = -\nabla p - \nabla \cdot \boldsymbol{\tau}^r, \end{cases} \tag{4}$$

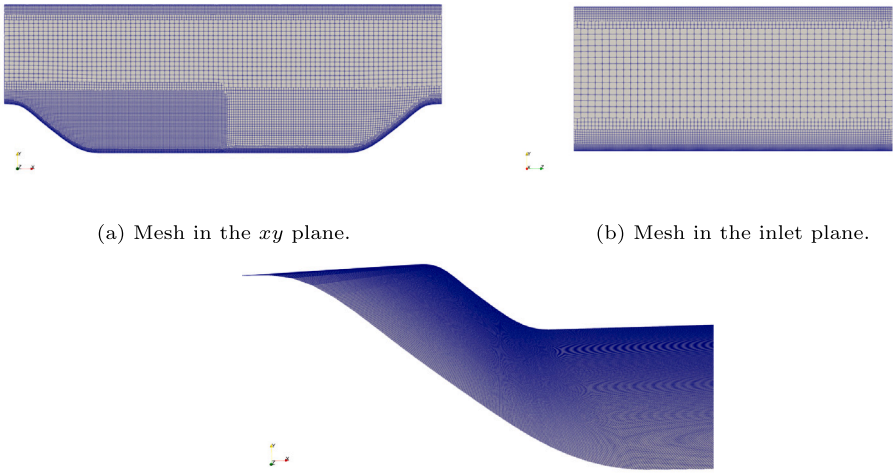
where, contrarily to the RANS equations, $\boldsymbol{u}$ and p are the velocity and pressure fields filtered in both time and space [33]. Because fields are not time-averaged and being turbulence an intrinsically time-dependent phenomenon, LES equations contain the derivative of the velocity field with respect to time even for time-averaged steady flows. The second order tensor $\boldsymbol{\tau}^r$ is the *residual stress tensor* and needs to be modeled. Here, we focus only on Smagorinsky sub-grid scale models [34] where the deviatoric part of $\boldsymbol{\tau}^r$ in modeled as

$$\boldsymbol{\tau}^r - \frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}^r)\boldsymbol{I} = -2\nu_t \boldsymbol{S}, \tag{5}$$

being $\boldsymbol{S}$ the symmetric part of the filtered velocity gradient and ν_t still a turbulent viscosity.

2.3. DES models

Detached eddy simulation can be interpreted as a RANS model that switches to a LES sub-grid scale model where turbulence length-scales exceed the computational grid dimensions. Usually, RANS model is used near wall while LES model is used far from wall. In this work we use a delayed detached eddy simulation (DDES) [35], where the transition from RANS to LES regions is performed through blending functions.



(a) Mesh in the xy plane.

(b) Mesh in the inlet plane.

(c) Mesh on the inlet hill wall.

Fig. 2. Computational mesh on different domain boundaries.

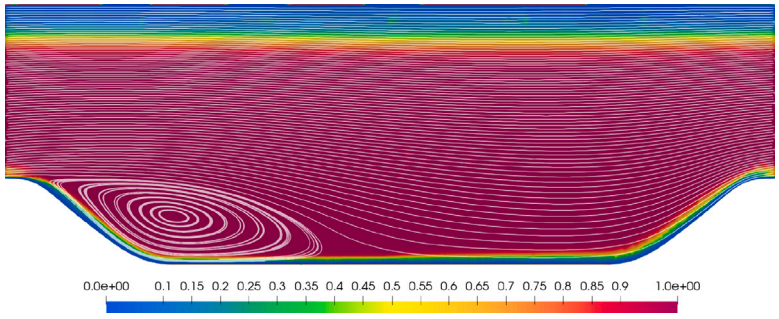


Fig. 3. Time average of turbulence models region splitting. Blue regions correspond to regions where only RANS has been used while in red regions where only LES has been used. Additionally, in white we show the streamlines of the time and spanwise averaged velocity field.

3. High fidelity simulations generation

The flow over periodic hills is a classic benchmark to access the performance of turbulence models, e.g. [36,37], thanks to availability of reliable both experimental and computational data [38–41] at different Reynolds numbers ranging from $Re = 700$ to $Re = 37000$. Among the benchmark references, Xiao et al. [23] built a dataset by performing DNS simulations over a class of parameterized geometries where hills steepness changes while keeping fixed the Reynolds number $Re = 5600$. This data availability drove researchers to use such flow configuration to test data-driven turbulence models [42–45].

To access the quality of the DDES simulations used in this work to generate our dataset, we describe the computational aspects and compare the results to the benchmark simulation [23] for the $\alpha = 1$ case.

3.1. Numerical set-up

The Reynolds number is $Re = 5600$ based on the hill height H and the bulk velocity U_b at the hill crest. Periodic boundary conditions are applied in the streamwise x and spanwise z directions and no-slip conditions are applied at wall. The computational domain has dimensions $L_x = 9H$ in the streamwise direction, $L_y = 3.036H$ in the y direction and $L_z = 4.5H$ in the spanwise direction following the recommendation in [39]. The mean flow is two-dimensional being the spanwise direction homogeneous.

The DDES equations have been discretized using a finite-volume based solver using a hexahedral mesh, see Fig. 2. The height of the first computational cell along the wall normal direction is $\Delta y_{hw}/H \approx 2\times10^{-3}$ on the hill wall and $\Delta y_{tw}/H \approx 4\times10^{-3}$ on the top wall. The expansion ratios across the wall-normal direction are 1.05 and 1.1 respectively. The computational mesh has been refined for $y/H < 1.2$ with a higher refinement for $x/H < 4.5$ where flow recirculation is expected, see streamlines in Fig. 3. The total number of computational cells is $N_{cells} \approx 5.9\times10^6$.

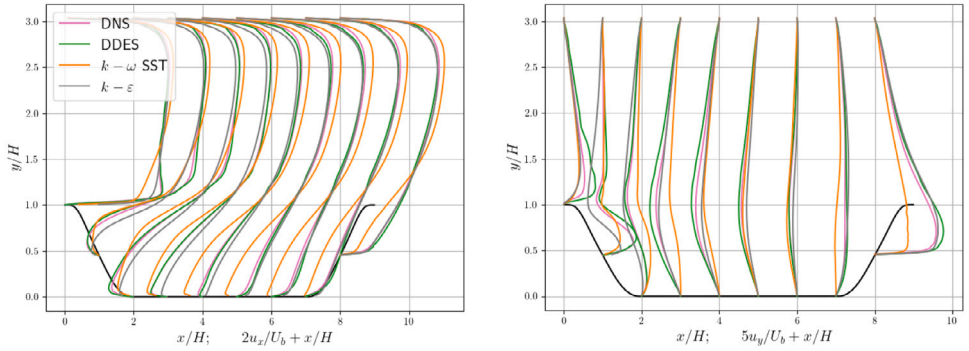
Table 1
Numerical parameters for DDES simulation with $\alpha = 1$.

Re	$\Delta y_{hw}/H$	$\Delta y_{tw}/H$	N_{cells}	$T_{avg}\ U_b/L_x$	$\Delta t\ U_b/L_x$
5600	2×10^{-3}	4×10^{-3}	5.9×10^6	40	4.5×10^{-4}

The flow is statistically steady and data are averaged after an initial transient state. Firstly, a RANS simulation is performed for 6.7 L_x/U_b and used as initial condition of the DDES simulation. The latter covers a time period $T = 60\ L_x/U_b$ and fields are averaged in the last $T_{avg} = 40\ L_x/U_b$. Statistical convergence has been checked by carrying a careful analysis on second order statistics such as Reynolds stresses. A time step of approximately $9\times10^{-4}\,L_x/U_b$ is initially adopted for the DDES run and decreased during the first stages of the transient period to attain at $t = 13\ L_x/U_b$ a value of $\Delta t \approx 4.5\times10^{-4}\ L_x/U_b$. Finally, the time-averaged fields are averaged in the spanwise direction too. Table 1 summarizes all the aforementioned parameters.

3.2. Validation to reference results

Firstly, in Fig. 3 we can observe in which regions the RANS model is prevalently used (blue regions) and in which LES is used (red regions).



(a) u_x velocity profiles (b) u_y velocity profiles

Fig. 4. Comparison of velocity profiles, being DNS profiles from Xiao et al. [23].

As expected from a DDES model, near wall the RANS model is used and it switches to LES moving far from wall.

To validate our results, we compare in Fig. 4 the velocity profiles with the benchmark case of in Xiao et al. [23]. In addition, we show the velocity profiles obtained using both the $k-\omega$ SST and $k-\varepsilon$ RANS models on a 2D mesh. The DDES result shows good agreement with the benchmark case, although significant discrepancies are observable for the vertical velocity, in particular close to the hills crests. On the other hand, among the two RANS models, the $k-\varepsilon$ model gives a better prediction of the recirculation region and of the vertical velocity but still poorly predicts the horizontal velocity near the bottom wall, whereas the $k-\omega$ SST model significantly overestimates the backward horizontal velocity in the bottom part of the domain and badly describes the vertical velocity across the domain.

Fig. 5 compares the Reynolds stresses profiles. RANS stresses have been computed starting from the k and either ε or ω fields using the Boussinesq's assumption. For both RANS models, profiles completely differ from the reference case while the DDES ones are closer. However, also in this case significant discrepancies can be observed, especially for the τ_{xy} component. It is worth mentioning that the Reynolds stresses will never be used in our machine learning approach, but their analysis are still a good metric to verify the effectiveness of our simulations and, in particular, the systematic improvement compared to the RANS models.

Finally, in Fig. 6 we present the distribution of y^+ at the hill wall where $y^+ = \Delta y_{hw} u_\tau / \nu$, being $u_\tau = \sqrt{\tau_{xy}/\rho}$ the shear stress velocity. The obtained y^+ is consistent with [23,39].

To conclude this section, we want to summarize the outcome of the analysis just carried out. DDES fields are in good agreement with respect to the results of Xiao et al. with some discrepancies. However, the latter results are obtained using high-order methods and significantly finer discretizations in both time and space to avoid any turbulence-scale modeling. We believe that DDES models can be useful to generate datasets if the main objective is to cover wider parametric spaces with significantly more simulations while still consistently improving the accuracy of standard RANS models, as found out in the above analysis.

4. Machine learning environment

4.1. Machine learning model

In this work we use the ν_t-Vector Basis Neural Network (ν_t-VBNN) as machine learning setting to close the RANS equations. This model, initially proposed in [10] and subsequently improved in [27], aims to predict through two feed-forward neural networks the turbulent viscosity ν_t and the component of the Reynolds force vector that is not taken into account in the turbulent viscosity term, defined as $\boldsymbol{t}^\perp = \nabla \cdot (\boldsymbol{\tau} + 2\nu_t \boldsymbol{S})$.

We deal with the Reynolds force vector (RFV) rather than the RST motivated by [13]. In particular, the RFV can be computed from scale-resolving simulations, like DNS or LES, using first-order statistics only. Consequently, if the time-averaging window is not wide enough, the RFV is less affected by errors compared to the Reynolds stresses, that are second-order statistics [14]. The choice to predict separately ν_t and $\boldsymbol{t}$ is driven by [11,12] where it is shown that this approach improves the conditioning of the new RANS system and it less affected by the intrinsic errors during the Machine-Learning regression step.

We opt to deal with $\tilde{\nu}_t = \nu_t/\nu$ and $\tilde{\boldsymbol{t}}^\perp = k^{1/2}/\varepsilon\ \boldsymbol{t}^\perp$ to handle dimensionless fields taking values of $O(1)$ or $O(10)$ and thus helping the learning procedure [46,47]. Following the ν_t-VBNN approach, $\tilde{\nu}_t$ is directly predicted by its neural network while $\tilde{\boldsymbol{t}}^\perp$ is written as linear combinations of a chosen vector basis and the learning targets of the training process are the coefficients of such combination

$$\tilde{\nu}_t = \nu_t(\lambda_1, \dots, \lambda_{N_i}), \tag{6a}$$

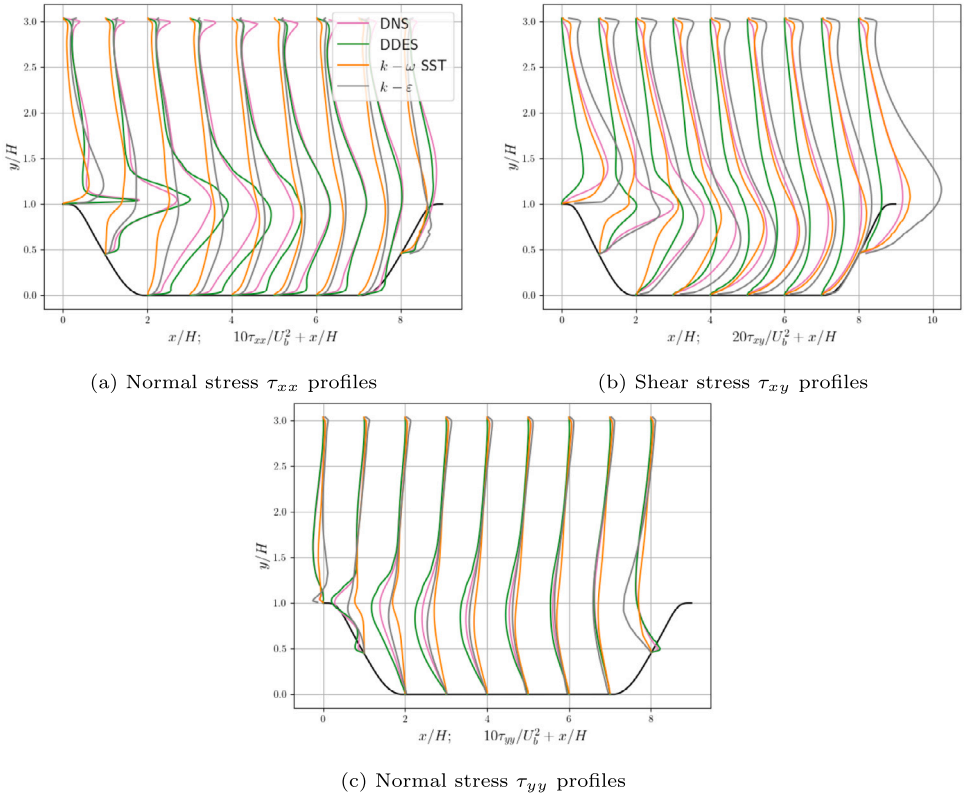
$$\tilde{\boldsymbol{t}}^\perp = \sum_{k=1}^{N_c} c_k(\lambda_1, \dots, \lambda_{N_i}) \boldsymbol{v}_k. \tag{6b}$$

Both $\tilde{\nu}_t$ and the coefficients c_k, $k = 1, \dots, N_c$, depend on scalar fields λ_i, $i = 1, \dots, N_i$, usually referred as *invariants*, obtained by appropriate multiplications of the symmetric and antisymmetric parts of the velocity gradient, $\boldsymbol{S}$ and $\boldsymbol{W}$ respectively, and ∇k. The expression of the invariants and of the vector basis is given in Appendix. The invariants are the inputs of the neural networks where the outputs are either the dimensionless turbulent viscosity or the coefficients of the $\tilde{\boldsymbol{t}}^\perp$ expansion. This particular setting enforces both Galilean invariance and independence of the chosen frame of coordinates. We refer to [10,27] for the discussion of the enforced physical properties of the ν_t-VBNN.

Finally, in this work we changed the wall-distance based Reynolds number $Re_d = \min(\frac{\sqrt{k}d}{50\nu}, 2)$ invariant [10,48,49] with $\frac{\sqrt{k}d}{2500\nu}$ to avoid discontinuities of its gradient. As a matter of fact, we observed that, when using the former in the training of $\tilde{\nu}_t$, nonphysical peaks of the predicted turbulent viscosity were obtained across the recirculation region where the gradient of Re_d is discontinuous.

4.2. Dataset

A total of 25 pairs of RANS-DDES simulations have been performed by spanning the geometrical parameter α defined in Section 3 to assume values $\alpha = 0.4, 0.45, \dots, 1.55, 1.6$. The $k-\omega$ SST model has been chosen as RANS model to challenge the ν_t-VBNN approach with the worst RANS model among the two investigated in the previous section. The RANS simulations have been performed on 2D meshes coarser than the DDES meshes projected in the xy-plane, see Fig. 7(a). Indeed, we observed that a coarser mesh could be used reaching already mesh convergence. The invariants are computed from the RANS simulations while



(a) Normal stress τ_{xx} profiles

(b) Shear stress τ_{xy} profiles

(c) Normal stress τ_{yy} profiles

Fig. 5. Comparison of Reynolds stresses profiles, being DNS profiles from Xiao et al. [23].

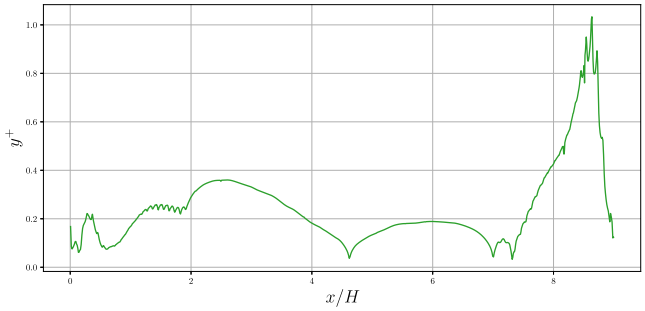


Fig. 6. y^+ distribution on the hill wall for $\alpha = 1$.

high-fidelity ν_t and $\boldsymbol{t}^\perp$ are obtained from the corresponding spanwise-averaged DDES simulations interpolated on the 2D RANS mesh. In particular, once ν_t is derived, the $\boldsymbol{t}^\perp$ field is computed indirectly [12] as

$$\boldsymbol{t}^\perp = -\boldsymbol{u}\nabla\boldsymbol{u} + \nabla\cdot[(\nu+\nu_t)\nabla\boldsymbol{u}] - \nabla p, \tag{7}$$

where $\boldsymbol{u}$ and p are the time averaged DDES fields. Finally, $\boldsymbol{t}^\perp$ is set dimensionless using the turbulent fields k and ε of the corresponding RANS simulations.

The obtained dataset consists of 4.51×10^5 computational cells. Fig. 7(b) shows the number of cells with respect to α, being that the higher is α, the larger the domain and, consequently, the more the computational cells.

4.3. Reference training settings

We define a reference training setting to be able to analyze in the second part of this section the effect of several training parameters. We will denote this case as `Ref`.

Data coming from all computational cells of every simulation except for the $\alpha = 1$ one are used for a total of 4.33×10^5 cells. Data are then split into 80% of them to train the two networks and 20% for validating only. The splitting into the two groups is random and independent for the two trainings.

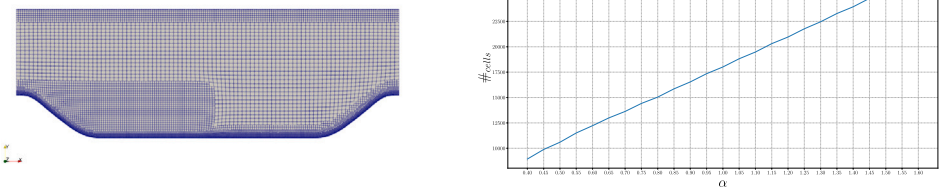
The two neural networks have 8 hidden layers made by 30 nodes each with ELU activation functions [50]. The two losses are the mean square errors with respect to the target $\tilde{\nu}_t$ and $\tilde{\boldsymbol{t}}^\perp$ with a L_2 regularization factor of 10^{-4} and 10^{-5} respectively. The trainings are performed using the pytorch library [51]. The Adam optimizer [52] is used with initial learning factors of 10^{-3} that decrease up to 10^{-6} during the trainings. Weights are updated using a mini-batching strategy with a batch size of 64. The maximum number of epochs per training is 500 but the training can be stopped earlier if the validation loss does not decrease for 80 consecutive epochs. For each network, 10 distinct trainings are performed and the best one in terms of validation error is kept.

5. Results

Once the ν_t-VBNN model is trained, both $\tilde{\nu}_t$ and $\tilde{\boldsymbol{t}}^\perp$ can be predicted using data coming from RANS simulations only, opportunely multiplied by powers of k and ε coming from the RANS and injected as fixed fields into the momentum Eq. (2) until new convergence of the RANS solver is reached.

To make the analysis more effective, we define an additional setting denoted by `Fix`. In this case, we insert into the RANS equations the ν_t and $\boldsymbol{t}^\perp$ of the DDES simulation interpolated on the RANS mesh without any data-driven model's approximation. It is an ideal setting of perfect data-driven model that does not make any error in the regression step and it represents the target accuracy that our data-driven model can attain.

In this section, we firstly investigate the velocity field obtained using the turbulent fields coming from the `Ref` setting and observe the better



(a) 2D RANS mesh for $\alpha = 1$. This mesh is coarser than the DDES mesh projected on the xy plane, Figure 2a.

(b) Number of computational cells with respect to α. The total cardinality of the dataset is 4.51×10^5.

Fig. 7. RANS mesh for $\alpha = 1$ (left) and RANS number of cells with respect to α (right).

Table 2
Reattachment points x_a/H.

DNS	DDES	RANS	Fix	Ref
4.64	4.61	7.60	4.26	4.65

results compared to the $k-\omega$ SST model. Successively, we investigate the effect of several training parameters by systematically changing them one-by-one.

5.1. Ref case

Fig. 8 compares the velocity field obtained using our DDES model, the $k-\omega$ SST model, the Fix case and our data-driven ν_t-VBNN model using the Ref setting discussed above. We also show the DNS velocity field to additionally test the accuracy of DDES results.

Regarding the u_x component, the $k-\omega$ SST model overestimates the values in the upper part of the domain and drastically overestimates the length of the recirculation region, being u_x still negative at the beginning of the second hill. Finally, it does not predict any maxima region close to the top of the hills' crests. On the other hand, the Ref case alleviates all RANS drawbacks. The discrepancies with respect to the DDES case (slight overestimation in the upper part of the domain and smaller maxima region on the top of hill crests) are observed in the Fix case too, letting suppose that could not be completely removed even with an ideal training setting.

Regarding the u_y component, again the RANS model shows important discrepancies with respect to the DDES case: the negative values in the middle of the domain are underestimated as for the maxima values after the first hill crest and before the downwinding hill crest. In general, the Ref field is close to both the Fix and the DDES cases.

In Fig. 9 we show the horizontal velocity at the first layer of cells attached to the hill wall. This quantity is useful to detect the reattachment point x_a/H corresponding to the x/H value with transition from negative to positive u_x. The exact transition value is reported in Table 2. The $k-\omega$ SST curve completely differs form the others and predict a late reattachment. On the other hand, the Ref curve predicts a reattachment point close to the DNS and DDES one and, in general, behaves like these two cases.

5.2. Analysis on some training parameters

We carry out a systematic analysis on some parameters regarding the training stage both in term of predicted ν_t and $\boldsymbol{t}^\perp$ fields and on resulting pressure and velocity fields.

To make an easier and more concise analysis, we define 6 different regions in the domain, shown in Fig. 10, and compute the relative errors of each field with respect to the DDES case. These regions are physically of interest: the Chan region focuses on the channel-like top part of the domain containing maxima values of u_x; analogously for ChanCut that focuses where u_x is minimum in Chan (see Fig. 8); Recirc corresponds to the recirculation region; RecircLong on the bottom part of the domain in general; ν_t−Max corresponds to the region of maximum of the turbulent viscosity; finally NearHill collects all computational cells near the hill wall depicted in Fig. 10(b). In the following, we denote by Ω the whole domain.

Table 3
Training settings for the different cases. In the "Same ν_t as Ref" column we report if the neural network for ν_t of the Ref setting is used or if a new one has been trained. $\alpha = 1$ is always excluded from training/validation dataset.

	α values	# cells	L_2-reg of $\tilde{\boldsymbol{t}}$	Same ν_t as Ref
Ref	all	4.33×10^5	10^{-5}	True
$\mathbf{t}^\perp$ − MoreReg	all	4.33×10^5	10^{-3}	True
CoarserRange	$0.4, 0.5, \ldots, 1.5, 1.6$	2.16×10^5	10^{-5}	False
SmallerRange	$0.7, 0.75, \ldots, 1.25, 1.3$	2.16×10^5	10^{-5}	False
lessData	all	2.04×10^5	10^{-5}	False

Given a training setting and a region $\mathcal{R}$, the errors for a scalar and vector field, s and $\boldsymbol{v}$ respectively, are defined as

$$e_s = \frac{\sum_{i\in\mathcal{R}} w^i \sqrt{(s^i - s^i_{DDES})^2}}{\sum_{i\in\mathcal{R}} w^i |s^i_{DDES}|}, \quad e_{\boldsymbol{v}} = \frac{\sum_{i\in\mathcal{R}} w^i \sqrt{\sum_{j=x,y} (v^i_j - v^i_{DDES,j})^2}}{\sum_{i\in\mathcal{R}} w^i \sqrt{\sum_{j=x,y} (v^i_{DDES,j})^2}}, \tag{8}$$

where the apex refers to the single computational cell in the $\mathcal{R}$ region and w^i is the volume of the i-th cell.

To make a fair comparison among pressure fields, we define the pressure coefficient

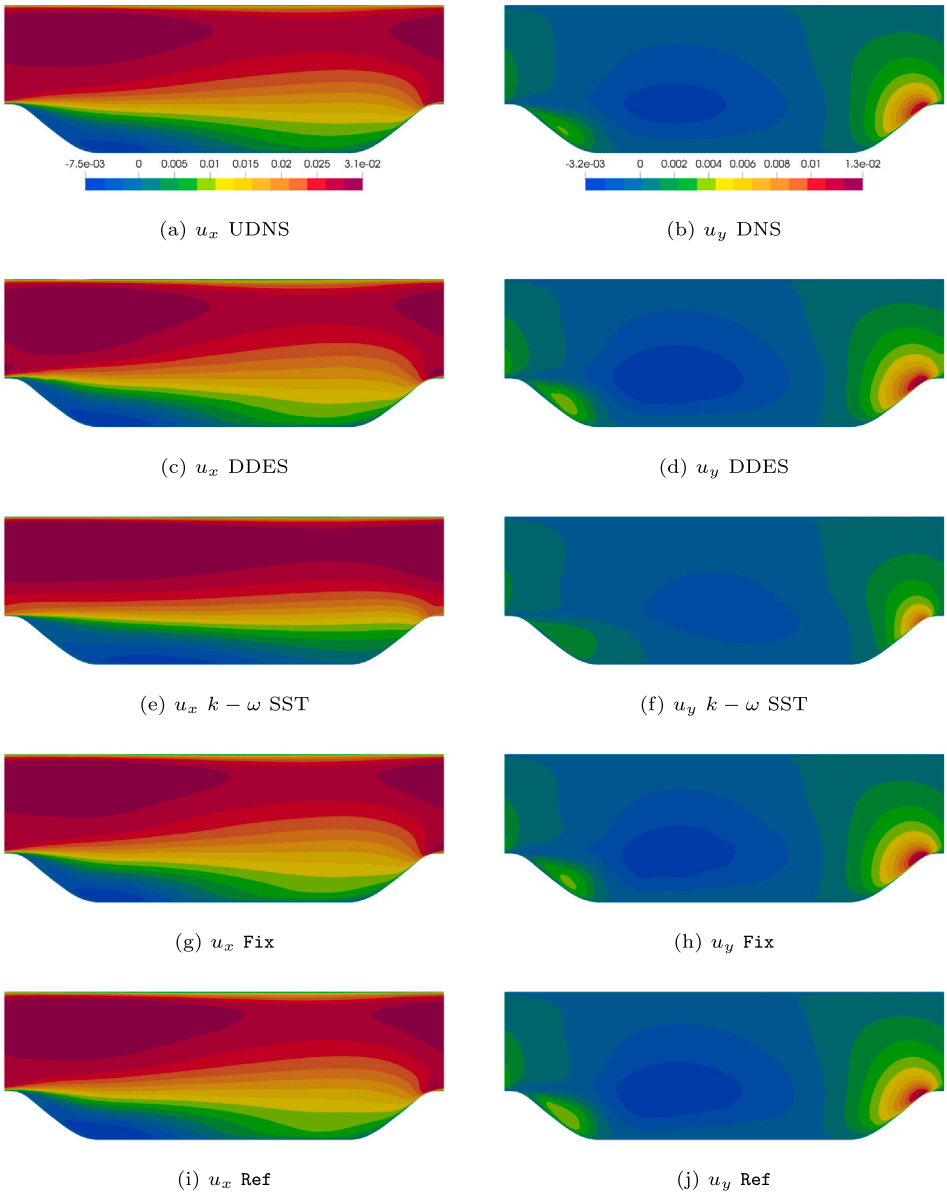
$$C_p = \frac{p - p_{ref}}{0.5\rho U_b^2}, \tag{9}$$

where ρ is the constant fluid density and p_{ref} is the pressure at the computational cell with coordinates $(0.051, 1.92)$, located near the first hill crest at middle channel height.

Finally, Table 3 summarizes all the cases that will be discussed in the following. For the training settings that involve changes on the $\tilde{\boldsymbol{t}}$ neural network only, we do not train a neural network for ν_t field and we use the Ref field instead. In all settings, data coming from the $\alpha = 1$ case are kept out from the training process.

5.2.1. Regularization factor in losses

We investigate the effect of increasing the L_2-regularization factor in the $\boldsymbol{t}^\perp$ loss function from 10^{-5} of the reference case to 10^{-3}. We denote this case by $\mathbf{t}^\perp$ − **MoreReg**. It is well known that the higher this parameter, the lower the risk of overfitting the training data but, at the same time, the higher the risk of underfitting unseen data. For this reason its value should be carefully chosen. The errors of the new $\mathbf{t}^\perp$ − **MoreReg** case with respect to the DDES case compared to the RANS



(a) u_x UDNS (b) u_y DNS

(c) u_x DDES (d) u_y DDES

(e) u_x $k-\omega$ SST (f) u_y $k-\omega$ SST

(g) u_x `Fix` (h) u_y `Fix`

(i) u_x `Ref` (j) u_y `Ref`

Fig. 8. u_x and u_y fields for DNS, DDES, RANS, `Fix` and `Ref` cases.

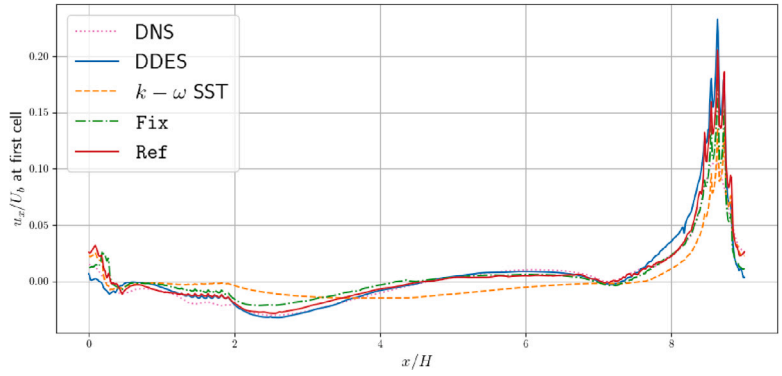
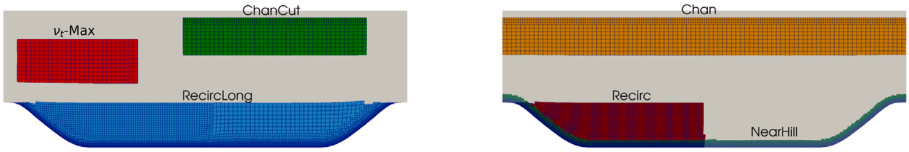


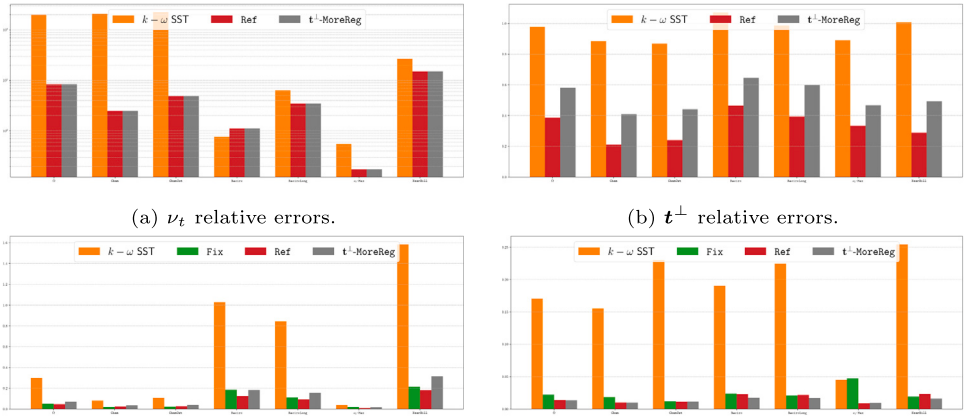
Fig. 9. u_x/U_b at the first layer of cells attached to the hill wall. Reattachment occurs where u_x goes from negative to positive.

and `Ref` errors are depicted in Fig. 11. Because we use the same ν_t field, the errors of the `Ref` and $\mathbf{t}^{\perp}$ – **`MoreReg`** are the same. We point out that the ν_t error plot has a logarithmic scale on the y-axis due to the high error for the RANS case. For all the remaining plots, a linear scale is adopted. When looking at the velocity and C_p fields, comparison with the `Fix` case is also included. To isolate the effect



(a) (b)

Fig. 10. Regions defined to investigate fields errors. They correspond to regions with important flow features.



(a) ν_t relative errors. (b) $\boldsymbol{t}^\perp$ relative errors.

(c) $\boldsymbol{u}$ relative errors. (d) C_p relative errors.

Fig. 11. Relative error in the domain and in the several regions for $\mathtt{t}^\perp$ – **MoreReg**.

of the $t^\perp$ regularization factor, we fix in the momentum equations the same turbulent viscosity of the `Ref` case.

It immediately appears in Fig. 11(b) that a value of 10^{-3} of the L_2-regularization term implies underfitting. As a matter of fact, the error for $t^\perp$ is significantly higher in all regions compared to the `Ref` one. While this behavior does not affect the accuracy of the C_p field, it increases the velocity error, particularly in the recirculation and near hill regions.

We carried the same analysis for the ν_t loss function and we noticed less systematic trends with respect to ν_t errors in the different regions and on resulting velocity errors. As a consequence, for the sake of brevity, we omit the corresponding error plots. This behavior justifies our conservative choice of setting a higher regularization factor of 10^{-4} for the `Ref` case.

5.2.2. α values during training

We analyze the role of the parametric choice in the dataset. Specifically, we investigate the case of a coarser training space with $\alpha = 0.4, 0.5, \dots, 1.5, 1.6$, denoted by `CoarserRange`, and of a smaller one with $\alpha = 0.7, 0.75, \dots, 1.25, 1.3$, denoted by `SmallerRange`. In both setting, the total datasets have cardinality of 2.16×10^5 cells (see Table 3).

Regarding the `CoarserRange` setting, Fig. 12, we observe that both learning targets ν_t and $t^\perp$ are generally less accurate. RANS-like errors are noticeable for the ν_t field in the `Chan` and `Chanut` regions while, in the other regions, the ν_t errors are slightly lower than the `Ref`. On the other hand, the $t^\perp$ error is systematically bigger that the reference case. These behaviors reflect on the worse description of both C_p and velocity in every region.

Regarding the `SmallerRange` setting, Fig. 13, we observe errors very close to the `Ref` ones for both ν_t and $t^\perp$, with a trend to slightly reduce it. This is expected because the neural networks are trained with a less variability of the data and are able to better specialize in interpolation cases like this one. However, no significant improvements can be noticed for the C_p and velocity field and, consequently, a more conservative approach of using a wider parameter interval is suggested even in an interpolation regime like this one.

5.2.3. Dataset cardinality

We investigate the role of the dataset cardinality by selecting randomly 8.5×10^3 cells from each α simulation ($\alpha = 1$ excluded) to create a new dataset. We call this setting `lessData`. In Fig. 14, we can observe that the learning targets are less accurate in most of the regions and equally accurate in fewer regions compared to the `Ref` case. The errors associated to C_p and $\mathbf{u}$ are consequently slightly higher.

As a final remark, we point out that one training epoch for the `Ref` case takes approximately twice the time of the `lessData` case, having the former a dataset that is roughly the double of the latter. Therefore, this approach could be beneficial when interested to reduce the training time if high accuracy is not foreseen.

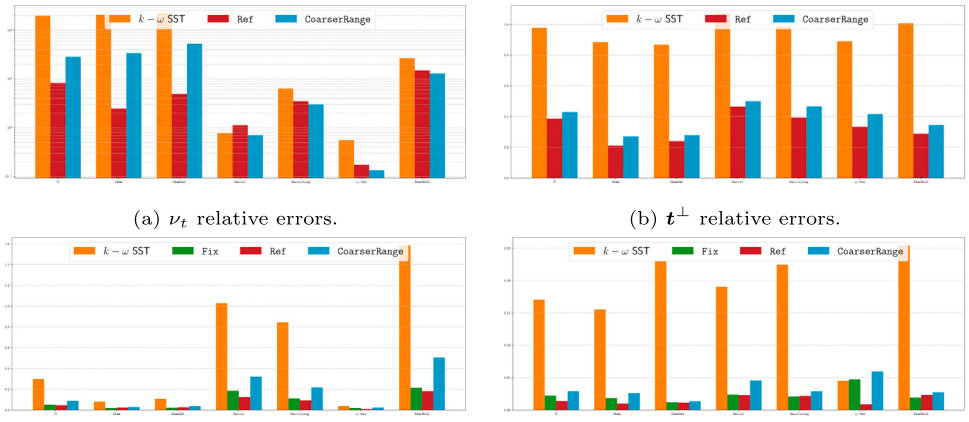
6. Conclusions

In this paper we investigate the possibility of using Delayed Detached Eddy Simulation (DDES) to create datasets for data-driven turbulence models. Specifically, we focus on the well-known and benchmarked Periodic Hills (PH) flow case and create a dataset consisting of 25 simulations with geometries parameterized by a factor α that determines the steepness of the hill's profiles. We cover a wide α range obtaining a considerable number of 4.51×10^5 computational cells.

We firstly check the accuracy of our simulations by carefully comparing the $\alpha = 1$ results with the DNS benchmark of Xiao et al. [23] and, successively, use them to train the ν_t-VBNN data-driven model [10,27].

We define a reference training setting and show the improvements compared to the baseline $k-\omega$ SST RANS model in terms of velocity field, reattachment point and pressure coefficient. We also investigate the effects of L_2-regularization in the losses, the choice of α values in the training phase and the dataset cardinality on the resulting fields both in all the domain and in physically-relevant regions.

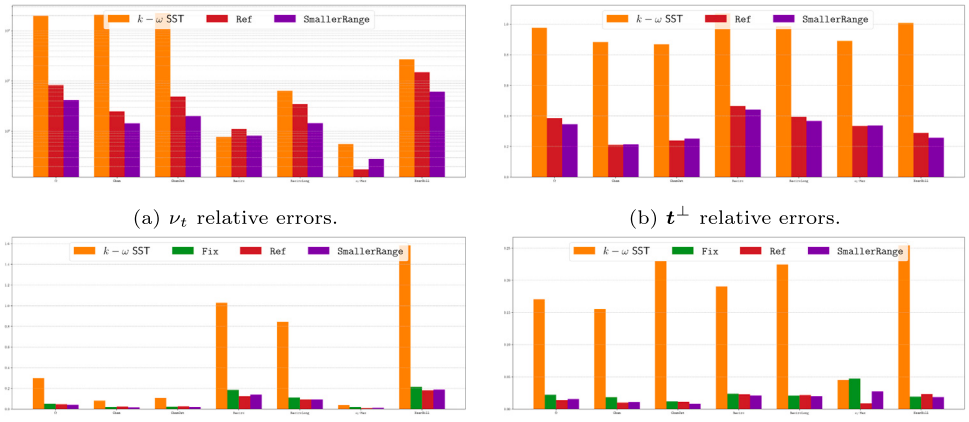
To summarize: despite generating high-fidelity data with DDES simulations instead of DNS or LES ones, we proved that the obtained dataset is reliable in terms of accuracy with respect to the DNS reference and effective to train data-driven turbulence models that signifi-



(a) ν_t relative errors. (b) $\boldsymbol{t}^{\perp}$ relative errors.

(c) $\boldsymbol{u}$ relative errors. (d) C_p relative errors.

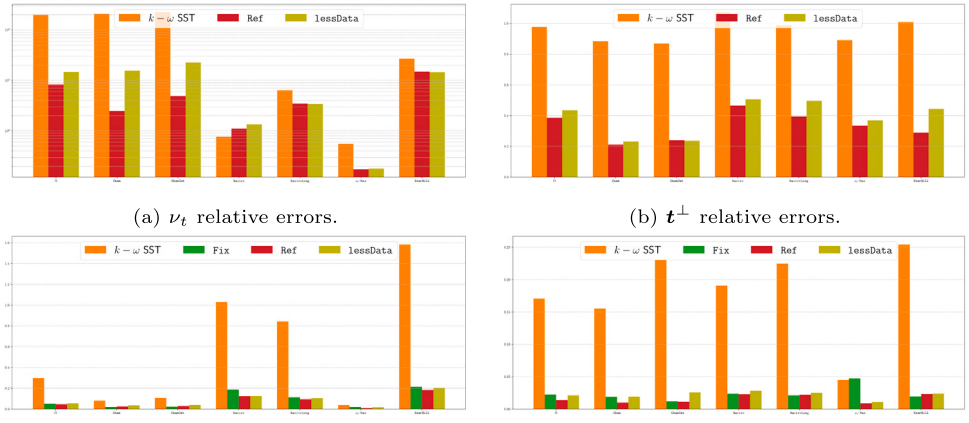
Fig. 12. Relative error in the domain and in the several regions for `CoarserRange`.



(a) ν_t relative errors. (b) $\boldsymbol{t}^{\perp}$ relative errors.

(c) $\boldsymbol{u}$ relative errors. (d) C_p relative errors.

Fig. 13. Relative error in the domain and in the several regions for `SmallerRange`.



(a) ν_t relative errors. (b) $\boldsymbol{t}^{\perp}$ relative errors.

(c) $\boldsymbol{u}$ relative errors. (d) C_p relative errors.

Fig. 14. Relative error in the domain and in the several regions for `less_data`.

cantly improves the classic $k-\omega$ SST RANS model. Using DDES models reduces significantly the computational cost of datasets generation, thus giving the opportunity of creating datasets spanning wider ranges of parameters.

The DDES results of this work are available on [28].

CRediT authorship contribution statement

Davide Oberto: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Davide Fransos:** Writing – review & editing, Supervision, Software, Data curation, Conceptualization. **Stefano Berrone:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Declaration of competing interest

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

Acknowledgments

D.O. and S.B. are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM) CUP_E53C23001670001. S.B. kindly acknowledge financial support by PNRR M4C2 project of CN00000013 National Centre for HPC, Big Data and Quantum Computing (HPC) CUP: E13C22000990001. The authors are grateful to Sauber Motorsport for hosting D.O. during part of his Ph.D. program.

Appendix. Invariants and vector basis

In [27], the following assumption is made:

$$\tilde{\boldsymbol{t}}^{\perp} = \tilde{\boldsymbol{t}}^{\perp}(\mathbf{s}, \mathbf{w}, \widetilde{\nabla \cdot \mathbf{S}}, \widetilde{\nabla k}, Re_d), \tag{A.1}$$

where $\mathbf{s} = \frac{k}{\epsilon}\boldsymbol{S}$, $\mathbf{w} = \frac{k}{\epsilon}\boldsymbol{W}$, $\widetilde{\nabla \cdot \mathbf{S}} = \frac{k^{5/2}}{\epsilon^2}\nabla \cdot \boldsymbol{S}$ and $\widetilde{\nabla k} = \frac{k^{1/2}}{\epsilon}\nabla k$ are the dimensionless counterparts of $\boldsymbol{S}$, $\boldsymbol{W}$, $\nabla \cdot \boldsymbol{S}$ and ∇k, respectively. Finally, $Re_d = \min(\frac{\sqrt{kd}}{50\nu}, 2)$ is the wall-distance based Reynolds number, where d is the wall distance.

With this assumption, we obtain (6b) [53], where the vector basis reads

$$\begin{aligned}
&\mathbf{v}_1 = \widetilde{\nabla \cdot \mathbf{S}}, \quad \mathbf{v}_2 = \mathbf{s}\ \widetilde{\nabla \cdot \mathbf{S}}, \quad \mathbf{v}_3 = \mathbf{s}^2\ \widetilde{\nabla \cdot \mathbf{S}},\\
&\mathbf{v}_4 = \mathbf{w}\ \widetilde{\nabla \cdot \mathbf{S}}, \quad \mathbf{v}_5 = \mathbf{w}^2\ \widetilde{\nabla \cdot \mathbf{S}}, \quad \mathbf{v}_6 = (\mathbf{sw} + \mathbf{ws})\ \widetilde{\nabla \cdot \mathbf{S}},\\
&\mathbf{v}_7 = \widetilde{\nabla k}, \quad \mathbf{v}_8 = \mathbf{s}\ \widetilde{\nabla k}, \quad \mathbf{v}_9 = \mathbf{s}^2\ \widetilde{\nabla k},\\
&\mathbf{v}_{10} = \mathbf{w}\ \widetilde{\nabla k}, \quad \mathbf{v}_{11} = \mathbf{w}^2\ \widetilde{\nabla k}, \quad \mathbf{v}_{12} = (\mathbf{sw} + \mathbf{ws})\ \widetilde{\nabla k},
\end{aligned} \tag{A.2}$$

while the invariants are

$$\begin{aligned}
&\lambda_1 = (\widetilde{\nabla \cdot \mathbf{S}})^T(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_2 = \mathrm{tr}(\mathbf{s}^2), \quad \lambda_3 = \mathrm{tr}(\mathbf{s}^3), \quad \lambda_4 = \mathrm{tr}(\mathbf{w}^2),\\
&\lambda_5 = \mathrm{tr}(\mathbf{sw}^2), \quad \lambda_6 = \mathrm{tr}(\mathbf{s}^2\mathbf{w}^2), \quad \lambda_7 = \mathrm{tr}(\mathbf{s}^2\mathbf{w}^2\mathbf{sw}), \quad \lambda_8 = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{s}(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_9 = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{s}^2(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_{10} = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{w}^2(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_{11} = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{sw}(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_{12} = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{s}^2\mathbf{w}(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_{13} = (\widetilde{\nabla \cdot \mathbf{S}})^T\mathbf{wsw}^2(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_{14} = (\widetilde{\nabla k})^T(\widetilde{\nabla k}), \quad \lambda_{15} = (\widetilde{\nabla k})^T\mathbf{s}(\widetilde{\nabla k}), \quad \lambda_{16} = (\widetilde{\nabla k})^T\mathbf{s}^2(\widetilde{\nabla k}),\\
&\lambda_{17} = (\widetilde{\nabla k})^T\mathbf{w}^2(\widetilde{\nabla k}), \quad \lambda_{18} = (\widetilde{\nabla k})^T\ \widetilde{\nabla \cdot \mathbf{S}}, \quad \lambda_{19} = (\widetilde{\nabla k})^T\mathbf{sw}(\widetilde{\nabla k}),\\
&\lambda_{20} = (\widetilde{\nabla k})^T\mathbf{s}^2\mathbf{w}(\widetilde{\nabla k}), \quad \lambda_{21} = (\widetilde{\nabla k})^T\mathbf{wsw}^2(\widetilde{\nabla k}), \quad \lambda_{22} = (\widetilde{\nabla k})^T\mathbf{sw}(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_{23} = (\widetilde{\nabla k})^T\mathbf{s}^2\mathbf{w}(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_{24} = (\widetilde{\nabla k})^T\mathbf{w}(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_{25} = (\widetilde{\nabla k})^T\mathbf{wsw}^2(\widetilde{\nabla \cdot \mathbf{S}}), \quad \lambda_{26} = (\widetilde{\nabla k})^T(\mathbf{sw} + \mathbf{ws})(\widetilde{\nabla \cdot \mathbf{S}}),\\
&\lambda_{27} = Re_d.
\end{aligned} \tag{A.3}$$

The first 26 invariants are obtained from the dependencies on $\mathbf{s}, \mathbf{w}$, $\widetilde{\nabla \cdot \mathbf{S}}, \widetilde{\nabla k}$. We observe that the invariant $\mathrm{tr}(\mathbf{s})$ is neglected because of the incompressibility constraint.

Finally, because $\tilde{\nu}_t$ is a scalar field, we directly use (6a) without involving any linear expansion.

Data availability

The data are shared through a GitHub link as indicated in the manuscript.

References

[1] Moin P, Kim J. Tackling turbulence with supercomputers. Sci Am 1997;276(1):62–8, ISSN: 00368733, 19467087, URL http://www.jstor.org/stable/24993565.
[2] Wilcox D. Turbulence modeling for CFD (Hardcover). 3rd ed.. Dcw Industries; 2006.
[3] Slotnick J, Khodadoust A, Alonso J, Darmofal D, Gropp W, Lurie E, et al. CFD vision 2030 study: A path to revolutionary computational aerosciences. Tech. rep., NASA; 2014.
[4] Kutz JN. Deep learning in fluid dynamics. J Fluid Mech 2017;814:1–4. http://dx.doi.org/10.1017/jfm.2016.803.
[5] Duraisamy K, Iaccarino G, Xiao H. Turbulence modeling in the age of data. Annu Rev Fluid Mech 2019;51(1):357–77. http://dx.doi.org/10.1146/annurev-fluid-010518-040547.
[6] Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech 2020;52(1):477–508. http://dx.doi.org/10.1146/annurev-fluid-010719-060214.
[7] Vinuesa R, Brunton SL. Enhancing computational fluid dynamics with machine learning. Nat Comput Sci 2022;2(6):358–66. http://dx.doi.org/10.1038/s43588-022-00264-7.
[8] Ling J, Jones R, Templeton J. Machine learning strategies for systems with invariance properties. J Comput Phys 2016;318:22–35. http://dx.doi.org/10.1016/j.jcp.2016.05.003.
[9] Ling J, Kurzawski A, Templeton J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J Fluid Mech 2016;807:155–66. http://dx.doi.org/10.1017/jfm.2016.615.
[10] Berrone S, Oberto D. An invariances-preserving vector basis neural network for the closure of Reynolds-averaged Navier–Stokes equations by the divergence of the Reynolds stress tensor. Phys Fluids 2022;34(9). http://dx.doi.org/10.1063/5.0104605.
[11] Wu J, Xiao H, Sun R, Wang Q. Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned. J Fluid Mech 2019;869:553–86. http://dx.doi.org/10.1017/jfm.2019.205.
[12] Brener BP, Cruz MA, Thompson RL, Anjos RP. Conditioning and accurate solutions of Reynolds average Navier–Stokes equations with data-driven turbulence closures. J Fluid Mech 2021;915:A110. http://dx.doi.org/10.1017/jfm.2021.148.
[13] Cruz MA, Thompson RL, Sampaio LE, Bacchi RD. The use of the Reynolds force vector in a physics informed machine learning approach for predictive turbulence modeling. Comput & Fluids 2019;192:104258. http://dx.doi.org/10.1016/j.compfluid.2019.104258, URL https://www.sciencedirect.com/science/article/pii/S0045793019302257.
[14] Thompson RL, Sampaio LEB, de Bragança Alves FA, Thais L, Mompean G. A methodology to evaluate statistical errors in DNS data of plane channel flows. Comput & Fluids 2016;130:1–7. http://dx.doi.org/10.1016/j.compfluid.2016.01.014, URL https://www.sciencedirect.com/science/article/pii/S0045793016300068.
[15] Cato AS, Volpiani PS, Mons V, Marquet O, Sipp D. Comparison of different data-assimilation approaches to augment RANS turbulence models. Comput & Fluids 2023;266:106054. http://dx.doi.org/10.1016/j.compfluid.2023.106054, URL https://www.sciencedirect.com/science/article/pii/S0045793023002797.
[16] Macedo MSS, Cruz MA, Brener BP, Thompson RL. A data-driven turbulence modeling for the Reynolds stress tensor transport equation. Internat J Numer Methods Fluids 2024. http://dx.doi.org/10.1002/fld.5284, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.5284, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.5284.
[17] Weatheritt J, Sandberg R. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. J Comput Phys 2016;325:22–37. http://dx.doi.org/10.1016/j.jcp.2016.08.015, URL https://www.sciencedirect.com/science/article/pii/S0021999116303643.
[18] Weatheritt J, Sandberg R. The development of algebraic stress models using a novel evolutionary algorithm. Int J Heat Fluid Flow 2017;68:298–318. http://dx.doi.org/10.1016/j.ijheatfluidflow.2017.09.017, URL https://www.sciencedirect.com/science/article/pii/S0142727X17303223.
[19] Schmelzer M, Dwight RP, Cinnella P. Discovery of algebraic Reynolds-stress models using sparse symbolic regression. Flow Turbul Combust 2020;104:579–603. http://dx.doi.org/10.1007/s10494-019-00089-x.
[20] Jiang C, Vinuesa R, Chen R, Mi J, Laima S, Li H. An interpretable framework of data-driven turbulence modeling using deep neural networks. Phys Fluids 2021;33(5):055133. http://dx.doi.org/10.1063/5.0048909.
[21] Zhou X-H, Han J, Xiao H. Frame-independent vector-cloud neural network for nonlocal constitutive modeling on arbitrary grids. Comput Methods Appl Mech Engrg 2022;388:114211. http://dx.doi.org/10.1016/j.cma.2021.114211.

[22] Volpiani PS. Are random forests better suited than neural networks to augment rans turbulence models? Int J Heat Fluid Flow 2024;107:109348. http://dx.doi.org/10.1016/j.ijheatfluidflow.2024.109348, URL https://www.sciencedirect.com/science/article/pii/S0142727X24000730.

[23] Xiao H, Wu J-L, Laizet S, Duan L. Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations. Comput & Fluids 2020;200:104431. http://dx.doi.org/10.1016/j.compfluid.2020.104431.

[24] Pinelli A, Uhlmann M, Sekimoto A, Kawahara G. Reynolds number dependence of mean flow structure in square duct turbulence. J Fluid Mech 2010;644:107–22. http://dx.doi.org/10.1017/s0022112009992242.

[25] Rapp C, Breuer M, Manhart M, Peller N. ERCOFTAC 2D periodic hill flow. 2010, https://kbwiki.ercoftac.org/w/index.php?title=Abstr:2D_Periodic_Hill_Flow.

[26] Spalart P, Deck S, Shur ML, Squiresand KD, Strelets MK, Travin A. A new version of detached-eddy simulation, resistant to ambiguous grid densities. Theor Comput Fluid Dyn 2006;20:181–95. http://dx.doi.org/10.1007/s00162-006-0015-0.

[27] Oberto D. Improving the vector basis neural network for RANS equations using separate trainings. 2024, http://dx.doi.org/10.48550/ARXIV.2409.17721.

[28] Oberto D, Fransos D, Stefano B. Flows over periodic hills using DDES turbulence models. 2024, https://github.com/DavideOberto/PeriodicHills_DDES_dataset.

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

[30] Fergizer JH, Peric M, Street RL. Computational method for fluid dynamic. Springer; 2020.

[31] Menter F. Zonal two equation k-w turbulence models for aerodynamic flows. In: 23rd fluid dynamics, plasmadynamics, and lasers conference. 2012, http://dx.doi.org/10.2514/6.1993-2906, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.1993-2906, URL https://arc.aiaa.org/doi/abs/10.2514/6.1993-2906.

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

[33] Leonard A. Energy cascade in large-eddy simulations of turbulent fluid flows. In: Frenkiel F, Munn R, editors. Turbulent diffusion in environmental pollution. Advances in geophysics, vol. 18, Elsevier; 1975, p. 237–48. http://dx.doi.org/10.1016/S0065-2687(08)60464-1, URL https://www.sciencedirect.com/science/article/pii/S0065268708604641.

[34] Smagorinsky J. General circulation experiments with the primitive equations: I. the basic experiment. Mon Weather Rev 1963;91(3):99–164. http://dx.doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2, URL https://journals.ametsoc.org/view/journals/mwre/91/3/1520-0493_1963_091_0099_gcewtp_2_3_co_2.xml.

[35] Gritskevich MS, Garbaruk AV, Schütze J, Menter FR. Development of DDES and IDDES formulations for the k-ω shear stress transport model. Flow Turbul Combust 2012;88:431–49. http://dx.doi.org/10.1007/s10494-011-9378-4.

[36] Jakirlic S, Jester-Zürker R, Tropea C. 9Th ERCOFTAC/IAHR/COST workshop on refined turbulence modelling, october 4-5, 2001. 2002, URL https://api.semanticscholar.org/CorpusID:127410514.

[37] Manceau R. Report on the 10th joint ERCOFTAC (SIG-15)/IAHR/QNET-CFD workshop on refined turbulence modelling, poitiers, october 10-11, 2002. ERCOFTAC Bull 2003;57:11–4, URL https://hal.science/hal-02991284.

[38] Fröhlich J, Mellen CP, Rodi W, Temmerman L, Leschziner MA. Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions. J Fluid Mech 2005;526:19–66. http://dx.doi.org/10.1017/S0022112004002812.

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

[40] Rapp C, Manhart M. Flow over periodic hills: an experimental study. Exp Fluids 2011;51:247–69. http://dx.doi.org/10.1007/s00348-011-1045-y.

[41] Gloerfelt X, Cinnella P. Large eddy simulation requirements for the flow over periodic hills. Flow Turbul Combust 2019;103:55–91. http://dx.doi.org/10.1007/s10494-018-0005-5.

[42] Amarloo A, Cinnella P, Iosifidis A, Forooghi P, Abkar M. Data-driven Reynolds stress models based on the frozen treatment of Reynolds stress tensor and Reynolds force vector. Phys Fluids 2023;35(7):075154. http://dx.doi.org/10.1063/5.0160977, arXiv:https://pubs.aip.org/aip/pof/article-pdf/doi/10.1063/5.0160977/18067105/075154_1_5.0160977.pdf, URL https://doi.org/10.1063/5.0160977.

[43] Amarloo A, Rincón MJ, Reclari M, Abkar M. Progressive augmentation of turbulence models for flow separation by multi-case computational fluid dynamics driven surrogate optimization. Phys Fluids 2023;35(12):125154. http://dx.doi.org/10.1063/5.0174470, arXiv:https://pubs.aip.org/aip/pof/article-pdf/doi/10.1063/5.0174470/18282594/125154_1_5.0174470.pdf, URL https://doi.org/10.1063/5.0174470.

[44] McConkey R, Kalia N, Yee E, Lien F-S. Turbo-RANS: straightforward and efficient Bayesian optimization of turbulence model coefficients. Internat J Numer Methods Heat Fluid Flow 2024. http://dx.doi.org/10.1108/hff-12-2023-0726.

[45] Brener BP, Cruz MA, Macedo MSS, Thompson RL. A highly accurate strategy for data-driven turbulence modeling. Comput Appl Math 2024;43:59. http://dx.doi.org/10.1007/s40314-023-02547-9.

[46] Ling J, Templeton J. Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. Phys Fluids 2015;27(8):085103. http://dx.doi.org/10.1063/1.4927765.

[47] Wu J-L, Xiao H, Paterson E. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. Phys Rev Fluids 2018;3(7):074602. http://dx.doi.org/10.1103/physrevfluids.3.074602.

[48] Wang J-X, Wu J-L, Xiao H. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. Phys Rev Fluids 2017;2(3):034603. http://dx.doi.org/10.1103/physrevfluids.2.034603.

[49] Milani PM, Ling J, Eaton JK. Turbulent scalar flux in inclined jets in crossflow: counter gradient transport and deep learning modelling. J Fluid Mech 2020;906. http://dx.doi.org/10.1017/jfm.2020.820.

[50] Clevert D-A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). 2015, arXiv:1511.07289.

[51] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, et al. Automatic differentiation in pytorch. In: NIPS-w. 2017.

[52] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv:1412.6980.

[53] Zheng Q-S. Theory of representations for tensor functions—a unified invariant approach to constitutive equations. Appl Mech Rev 1994;47(11):545–87. http://dx.doi.org/10.1115/1.3111066.