Progressive, extrapolative machine learning for near-wall turbulence modeling

Yuanwei Bin,[1,2] Lihua Chen,[3] George Huang,[4,*] and Xiang I. A. Yang[1,†]

[1]*Department of Mechanical Engineering, Pennsylvania State University, State College, Pennsylvania 16802, USA*

[2]*State Key Laboratory for Turbulence and Complex Systems, Peking University, Beijing 100871, China*

[3]*Department of Engineering Mechanics, Zhejiang University, Hangzhou 310027, China*

[4]*Department of Mechanical and Materials Engineering, Wright State University, Dayton, Ohio 45435, USA*

(Received 15 March 2022; accepted 15 August 2022; published 26 August 2022)

Conventional empirical turbulence modeling is progressive: one begins by modeling simple flows and progressively works towards more complex ones. The outcome is a series of nested models, with the next, more complex model accounting for some additional physics relative to the previous, less complex model. The above, however, is not the philosophy of data-enabled turbulence modeling. Data-enabled modeling is one stop: one trains against a group of data, which contains simple and complex flows. The resulting model is the best fit of the training data but does not closely reproduce any particular flow. The differences between the two modeling approaches have left data-enabled models open to criticism: machine learned models do not fully preserve, e.g., the law of the wall (among other empirical facts), and they do not generalize to, e.g., high Reynolds numbers (among other conditions). The purpose of this paper is to respond to and resolve some of these criticisms: we intend to show that the conventional progressive modeling is compatible with data-enabled modeling. The paper hinges on the extrapolation theorem and the neutral neural network theorem. The extrapolation theorem allows us to control a network's behavior when extrapolating and the neutral neural network theorem allows us to augment a network without "catastrophic forgetting." For demonstration purposes, we successively model the flow in the constant stress layer, which is simple; the flow in a channel and a boundary layer, which is more complex; and wall-bounded flow with system rotation, which is even more complex. We show that the more complex models respect the less complex models, and that the models preserve the known empiricism.

DOI: 10.1103/PhysRevFluids.7.084610

I. BACKGROUND AND MOTIVATION

Resolving all the scales remains prohibitively costly for flows at high, practically relevant Reynolds numbers [1–3], and one has to rely on turbulence models for predictive modeling in the foreseeable future. We consider two approaches to turbulence modeling, namely, the conventional empirical modeling approach [4–8] and the more recent data-enabled approach [9–12]. Empirical turbulence modeling relies heavily on dimensional arguments, intuitions, and empiricism. Data play a secondary role and are invoked only to determine model constants.

[*]george.huang@wright.edu
[†]xzy48@psu.edu

The approach dates back to Prandtl and his mixing length model and has since accumulated a large user base [13]. On the other hand, data-enabled modeling gained popularity in the past five years, although the first use of machine learning in turbulence modeling dates back to at least [14,15] in the 1990s and early 2000s. Here, we have adopted the terms "conventional empirical models" and "data-enabled models" rather than the more colloquial terms "physics-based models" and "data-based models." The latter is problematic because physics is a building block of both conventional physics-based and data-enabled models. The two modeling approaches have rather different underlying logic and their strengths and weaknesses. In the following, we explain, in general terms, what are empirical turbulence modeling and data-enabled turbulence modeling.

Conventional empirical modeling is progressive: one begins by modeling a simple, tractable flow and then more complex flows. Consider, e.g., boundary-layer flows. The beginning of boundary-layer modeling is the mixing length model (1925) [16,17],

$$\nu_t = \kappa y u_\tau D, \;\; D = [1 - \exp(-y^+/A^+)]^2, \tag{1}$$

then the wake layer (1956) [18],

$$\nu_t = \min[\kappa y u_\tau D, \; \alpha \delta^* U_0], \tag{2}$$

then boundary layers with system rotation (2020) [19]:

$$\nu_t = \min[\kappa y u_\tau D, \; u_\tau^2/(2\Omega)]. \tag{3}$$

The end result is a series of nested models as shown in Eqs. (1), (2), and (3), with the next, more complex model accounting for some additional physics relative to the previous, less complex model. Here, ν_t is the eddy viscosity, u_τ is the friction velocity, D is the damping, A^+ is the damping constant, y is the wall-normal coordinate, κ is the von Kármán constant, α is a constant, δ^* is the displacement height, U_0 is the freestream velocity, and Ω is the spanwise system rotation. Data-enabled turbulence modeling is one stop: one trains a model against a group of data in one sitting—be it *a priori* [20–24] or model consistent [25–28]. The training data set may contain simple flows like channel, more complex separated flows, and very complex flows like in [29,30]. The resulting model is a compromise between these flows and does not "fully" respect any one flow. Adding new training data changes the resulting model and its performance in previously existing flows. Take the model in [31] as an example. The model is a compromise of the flows in the training dataset, but it does not respect one flow, e.g., channel, more than any other flow, e.g., jet flow. As a result, the model does not preserve the law of the wall or extrapolate to unseen flow conditions. In all, data-enabled turbulence modeling and empirical modeling have different philosophies: the latter aims to develop a general model, while the former aims to train an accurate model for a limited number of flows.

Trading off generality for accuracy is, in principle, one's choice. However, computational fluid dynamics is a field where getting training data is costly, and practitioners need to handle unseen flows. These circumstances make trading off generality for accuracy undesirable. This is exacerbated as empirical modeling has a large experience base. Not following the conventional empirical modeling approach has left data-enabled turbulence modeling open to criticisms. Spalart [32] pointed out that proposing a machine learned model for a specific flow is instructive, but the model is not a product. He also pointed out that a successful data-enabled model should preserve the law of the wall and be open to corrections for other physics such as curvature, compressibility, and anisotropy. The purpose of this paper is to (partly) address these criticisms.

The rest of the paper is organized as follows. We review the basics of artificial neural networks and present the general framework of progressive machine learning in Sec. II. We show examples of progressive data-enabled turbulence modeling in Sec. III and conclude in Sec. IV.

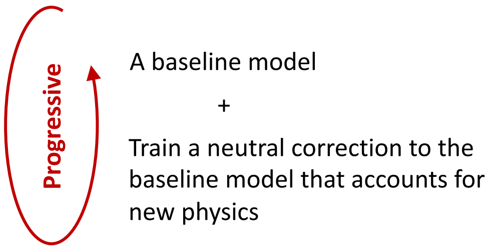
Progressive
A baseline model
+
Train a neutral correction to the baseline model that accounts for new physics

FIG. 1. A sketch of the modeling framework. The modeling philosophy is progressive.

II. METHODOLOGY

A. Theoretical foundation

In this section, we name two straightforward mathematical theorems. They will be the theoretical foundation of progressive machine learning. We will limit the discussion to single-layer feedforward neural networks. Nonetheless, because of the universal approximation theorem, the discussion here should apply equally to deep neural networks, as we will verify empirically in Sec. III.

Extrapolation theorem. For a bias-free (no bias units) nontrivial neural network that maps from $\mathbb{R}^1$ to $\mathbb{R}^1$, we have

$$\text{net}(\infty) = \text{a finite constant} \tag{4}$$

if we employ the sigmoidal transfer function for all neurons, and

$$\lim_{x\to\infty} \text{net}(x) \sim x \tag{5}$$

if we employ the rectified linear unit as the transfer function for all neurons. Here, "net" is a feedforward neural network.

Detailed proof is not included here for brevity. This theorem enables us to control how a neural network extrapolates. Consider, e.g., training a network to model the mixing length in the log layer. The training data are available at finite Reynolds numbers, but we want the network to extrapolate to infinitely large y^+ such that the asymptotic behavior is $l_m^+ \sim y^+$. The extrapolation theorem guarantees the following: first, a ReLu-activated neural network that takes y^+ as its input and gives l_m^+ as its output will have the correct asymptotic behavior at the infinite Reynolds number; second, a sigmoidal activated neural network that takes y^+ as its input and gives l_m/y as its output will have the correct asymptotic behavior at the infinite Reynolds number.

Neutral neural network theorem. Zero input guarantees zero output if we remove all bias units in a fully connected single-layer feedforward neural network.

Again, detailed proof is not included here for brevity. This theorem enables us to progressively improve an existing neural network without breaking its original behavior.

B. Framework of progressive machine learning

Figure 1 is a sketch of the general framework. The modeling is progressive. Every iteration will start from a baseline model and arrive at a neutral correction to that baseline model. The correction accounts for some additional physics and is neutral when the physics it accounts for is absent.

Specifically, we begin with a baseline model MODEL that accounts for some basic physics. The physics is parametrized in $\mathbf{x}$ (be it compressibility, Reynolds number, etc.). We then parametrize the additional physics in a nondimensional vector $\mathbf{X}$. The vector $\mathbf{X}$ is $\mathbf{0}$ when the additional physics is absent. Then, we train a bias-free neural network CORRECTION as a correction to MODEL. CORRECTION takes $\mathbf{x}||\mathbf{X}||_2$ and $\mathbf{X}$ as its input, where $||\cdot||_2$ is the L2 norm. The neutral neural

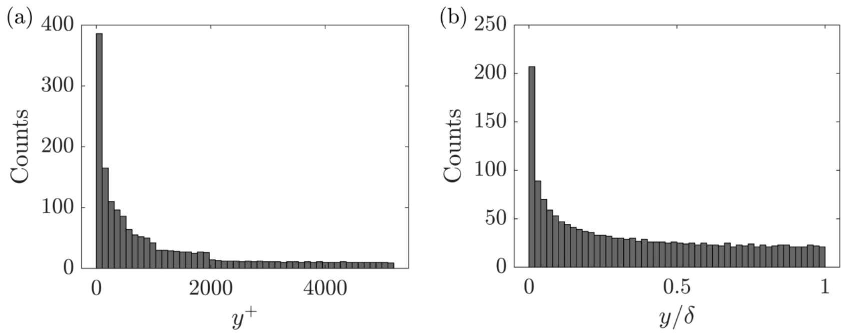

FIG. 2. Counts of samples if one gathers channel flow direct numerical simulation in [34] without any preprocessing. The Reynolds numbers are $\mathrm{Re}_\tau = 180, 550, 1000, 2000,$ and 5200. (a) Counts of samples in the y^+ space. (b) Counts of samples in the y/h space.

network theorem dictates that CORRECTION returns zero when $||\mathbf{X}||_2$ is zero. The resulting model

$$\mathrm{MODEL}^*(\mathbf{x}, \mathbf{X}) = \mathrm{MODEL}(\mathbf{x}) + \mathrm{CORRECTION}(\mathbf{x}||\mathbf{X}||_2, \mathbf{X})$$

accounts for some more physics than MODEL and meanwhile protects learned physics in MODEL. That is, MODEL*= MODEL when $\mathbf{X} = \mathbf{0}$. Here, it is worth noting that a function of $\mathbf{x}$ and $\mathbf{X}$ is still a function of $\mathbf{x}||\mathbf{X}||$ and $\mathbf{X}$ and vice versa, and therefore the process is general.

C. Preparing training data: A caveat

The physical space is continuous, and one can cluster data however one wants. As a result, training data preparation hosts much arbitrariness. This is an important but often overlooked issue in machine learning. To explain this issue, we consider eddy viscosity modeling in a channel. We know

$$\nu_t^+ = f_1(y^+) + f_2(y/h), \tag{6}$$

in a channel. Here, f_1 and f_2 are functions of y^+ and y/h, respectively. Here, h is the half channel height. Ideally, we would want to sample the y^+ space evenly or the $\log(y^+)$ space such that there are more samples on the response surface where the gradient is large and vice versa.

If one gathers channel flow data from the online repositories [33,34] without any preprocessing, the wall layer is weighed more than the wake layer. Figures 2(a) and 2(b) show the sample counts in the y^+ and the y/h spaces. The sample count is a decreasing function of y^+ and y/h because the grid clusters at the wall and because high y^+ value appears only at high Reynolds numbers. Following the discussion above, we ought to resample the data before any training.

III. EXAMPLES OF PROGRESSIVE MACHINE LEARNING

We work through a few examples. The purpose is to illustrate progressive machine learning. We will train neural networks to model the eddy viscosity in the constant-stress layer, the wake layer, and a rotating channel. The networks are single-layer, fully connected, feedforward neural networks that contain fewer than 20 neurons in the hidden layer. The tanh unit is employed as the transfer function. Again, because of the universal approximation theorem, any conclusion here applies to deep feedforward neural networks and vice versa. The neural network predicted eddy viscosity is then employed to close the momentum equation and solve for the velocity.

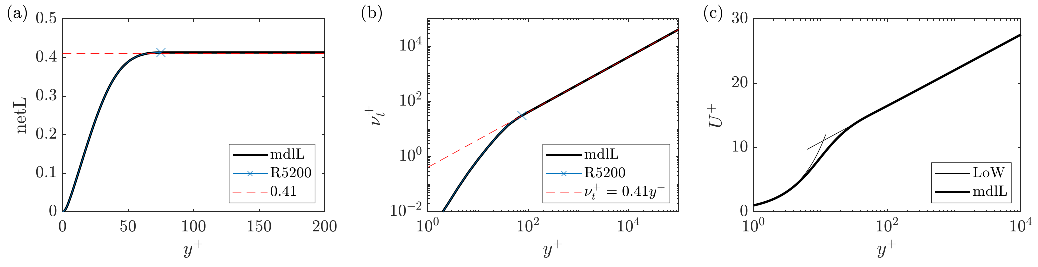

FIG. 3. (a) ν_t^+/y^+ as a function of y^+. Shown here are netL, training data at $\mathrm{Re}_\tau = 5200$, i.e., R5200 in the figure, and the expected asymptote, i.e., $\kappa \approx 0.41$. The training data at other lower Reynolds numbers are not shown here as they cover a narrower y^+ range. The training data cover y^+ up to 74. In the figure, the cross symbol is at $y^+ = 74$. The x axis is in linear scale and goes to $y^+ = 200$. (b) The eddy viscosity ν_t^+ as a function of y^+ in log-log scale. Shown here are mdlL, the training data at $\mathrm{Re}_\tau = 5200$, i.e., R5200 in the figure, and the expected scaling in the logarithmic layer, i.e., $\nu_t^+ = 0.41y^+$. The x axis goes up to $y^+ = 10^5$. (c) The velocity profile. LoW corresponds to the linear scaling in the viscous sublayer and the logarithmic scaling in the logarithmic layer. (a, b) *A priori* tests. (c) *A posteriori* test.

A. Logarithmic layer

We train an eddy viscosity for the logarithmic layer. This is the first iteration, and we do not have a baseline model from the previous iteration. In other words, the baseline model is

$$\mathrm{MODEL} = 0. \tag{7}$$

The objective is to train a CORRECTION to account for the log layer physics. Per the extrapolation theorem, the known asymptote $l_m \sim y$ will be preserved, if we train a sigmoid activated fully connected feedforward neural network to learn ν_t^+/y^+ as a function of y^+:

$$\nu_t^+/y^+ = \mathrm{CORRECTION}(y^+). \tag{8}$$

Here, y^+ is the viscous unit scaled distance from the wall, and $\nu_t = -\langle u'v'\rangle/(dU/dy)$ is the eddy viscosity. We make use of the channel flow direct numerical simulation (DNS) data in [34] for training. The training data are limited to $y/h < 0.015$—any further into the bulk, the outer length scale starts to play a noticeable role. The trained neural network is referred to as netL. The model is referred to as mdlL.

Figures 3(a) and 3(b) compare mdlL, the training data, and the known asymptote. The training data cover y^+ up to 74. There, ν_t^+/y^+ is already at its asymptote, beyond which ν_t^+/y^+ is a constant. As expected, a sigmoid activated fully connected feedforward network has no problem learning this asymptotic behavior: netL follows the training data up to $y^+ = 74$, beyond which it stays a constant. The ability to extrapolate is more clearly shown in Fig. 3(b), where y^+ goes up to 10^5 and mdlL gives the correct asymptotic behavior. Figure 3(c) shows the velocity profile as a function of y^+. The mdlL result follows closely the law of the wall and extrapolates to unseen Reynolds numbers.

B. Channel flow

For our second iteration, we build on netL (a baseline model) and train a network for the entire channel. The new physics is the wake layer physics. The half channel height h enters as a new input, y/h. We train a bias-(unit)-free neural network netC such that

$$\nu_t^+/y^+ = \mathrm{netL}(y^+) + \mathrm{netC}(y^+(y/h), y/h). \tag{9}$$

The resulting model and all its results are denoted as mdlC. In the logarithmic layer, $\lim_{\mathrm{Re}_\tau\to\infty} y/h \to 0$ whereas y^+ is finite. In this limit netC $\to 0$, and the new model mdlC should respect the baseline model mdlL as a result of the neutral neural network theorem—which we will verify. The training data are still the channel flow DNSs in [34], but we resample such that the training data sample the y/h space evenly.

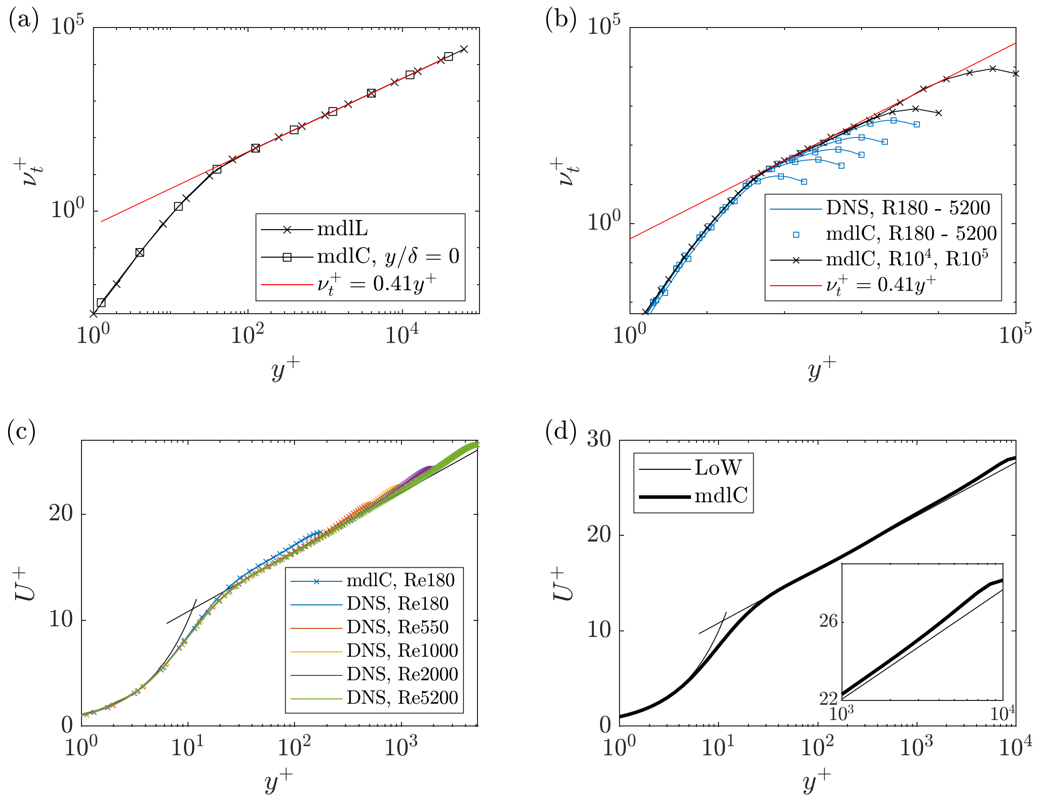

FIG. 4. (a) mdlL and mdlC predicted eddy viscosity as a function of the viscous scaled wall-normal distance y^+ for finite y^+, $y/h \to 0$, $\mathrm{Re}_\tau \to \infty$. (b) mdlC predicted eddy viscosity. The blue lines are the training data. The blue symbols are the mdlC results at the conditions in the training dataset. The $\times$ symbols (connected by lines) are mdlC results at a Reynolds number higher than those in the training dataset. (c) Mean velocity U^+ as a function of the viscous scaled wall-normal distance y^+ for $\mathrm{Re}_\tau \leqslant 5200$. The lines are DNS results. The symbols are mdlC results. The thin black lines represent the linear scaling in the viscous sublayer and the logarithmic scaling in the logarithmic layer. (d) Same as (c) but for $\mathrm{Re}_\tau = 10^4$. The inset is a zoom in view of the velocity profile in the wake layer. (a, b) *A priori* tests. (c, d) *A posteriori* tests.

Figures 4(a) and 4(b) show the eddy viscosity ν_t^+ as a function of y^+. Figure 4(a) compares mdlL and mdlC for $\mathrm{Re}_\tau \to \infty$, $y/h \to 0$. We see that mdlL $\equiv$ mdlC when $y/h \to 0$: the more complex model mdlC fully respects the simple model mdlL. Figure 4(b) compares mdlC at $\mathrm{Re}_\tau = 180$ to 5200 (inside the training data) and at $\mathrm{Re}_\tau = 10^4$ and 10^5 (outside the training data). We see that mdlC follows the training data very closely and extrapolates well to higher Reynolds numbers. Figures 4(c) and 4(d) show the velocity profile U^+ as a function of y^+. Figure 4(c) compares the mdlC results with the DNS for $\mathrm{Re}_\tau \leqslant 5200$, and Fig. 4(d) shows the mdlC result at $\mathrm{Re}_\tau = 10^4$. We see that mdlC follows the training DNS data very closely in Fig. 4(c). We also see the mdlC extrapolates well in Fig. 4(d).

C. Boundary-layer flow

We can repeat the exercise in Sec. III B for zero-pressure-gradient boundary-layer flows and train a network netB such that

$$\nu_t/y^+ = \mathrm{netL}(y^+) + \mathrm{netB}(y^+(y/\theta), y/\theta), \tag{10}$$

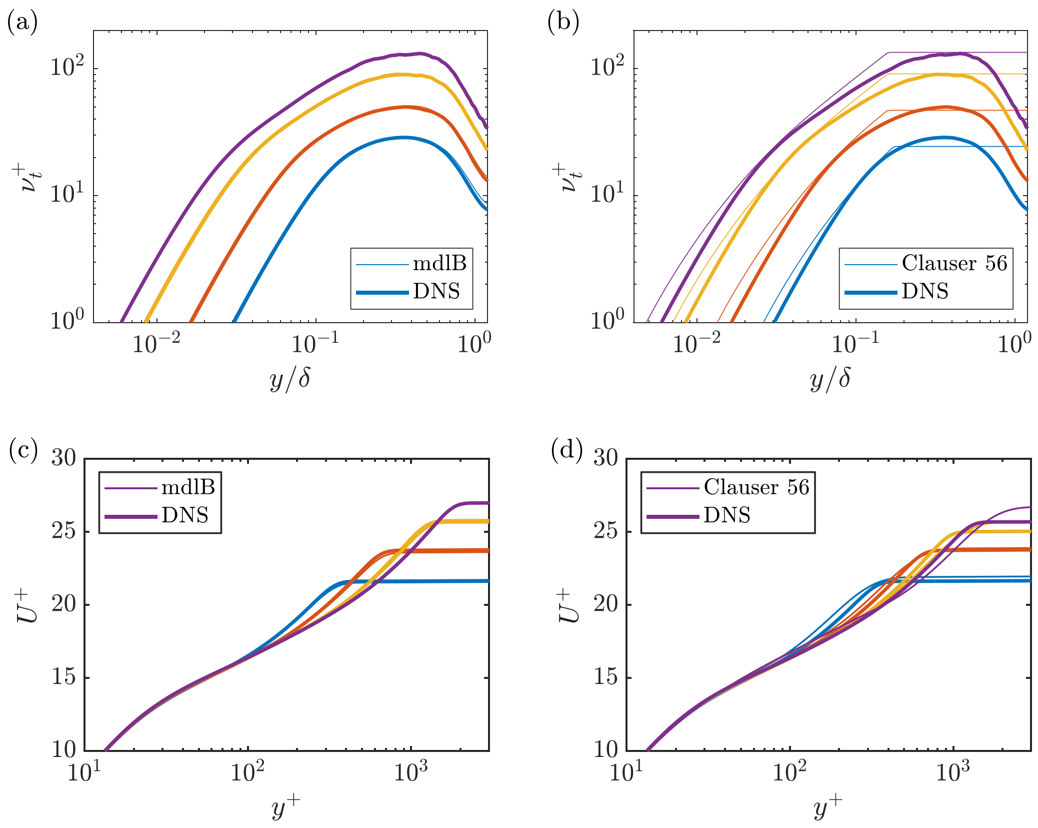

FIG. 5. (a, b) Eddy viscosity ν_t^+ as a function of y/δ at $\mathrm{Re}_\theta = 1000$, 2000, 4000, and 6000. (c, d) Velocity profiles U^+ as a function of y^+ at $\mathrm{Re}_\theta = 1000$, 2000, 4000, and 6000. (a, c) A comparison between mdlB and DNS. (b, d) A comparison between the empirical model [18] and DNS. The blue lines are for $\mathrm{Re}_\theta = 1000$, the red lines are for $\mathrm{Re}_\theta = 2000$, the yellow lines are for $\mathrm{Re}_\theta = 4000$, and the purple lines are for $\mathrm{Re}_\theta = 6000$. (a, b) *A priori* tests. (c, d) *A posteriori* tests.

where θ is the momentum thickness and is an outer length scale. The training data are readily available in [35], and the Reynolds number is between $\mathrm{Re}_\theta = 670$ and 6500, where $\mathrm{Re}_\theta = \theta U_\infty/\nu$, and θ is the momentum thickness. We preprocess the training data to sample the y/θ space evenly. The eddy viscosity is undefined in the freestream (since there is no turbulence) and we cut off at $y/\delta_{99} = 1.0$, where the velocity U is 99% of the freestream velocity.

In the following, we compare mdlB to the empirical model Eq. (2) [18]. Figures 5(a) and 5(b) show the eddy viscosity, and Figs. 5(c) and 5(d) show the velocity profiles. The machine learning model mdlB is more accurate than the empirical model in its predictions of both the eddy viscosity and the velocity. This is not very surprising since mdlB is trained to fit the DNS data. We are yet to confirm if mdlB extrapolates. Figure 6(a) shows the skin friction coefficient $C_f = 2\tau_w/\rho U_\infty^2$, where U_∞ is the freestream velocity, and Fig. 6(b) shows the parameter Π in Cole's law of the wake, where

$$U^+ = \frac{1}{\kappa}\log(y^+) + B + \frac{2\Pi}{\kappa} f(y/\delta) \tag{11}$$

and

$$\Pi = \frac{\kappa}{2}\left[U_\infty^+ - \frac{1}{\kappa}\log(\delta_{99}^+) - B\right]. \tag{12}$$

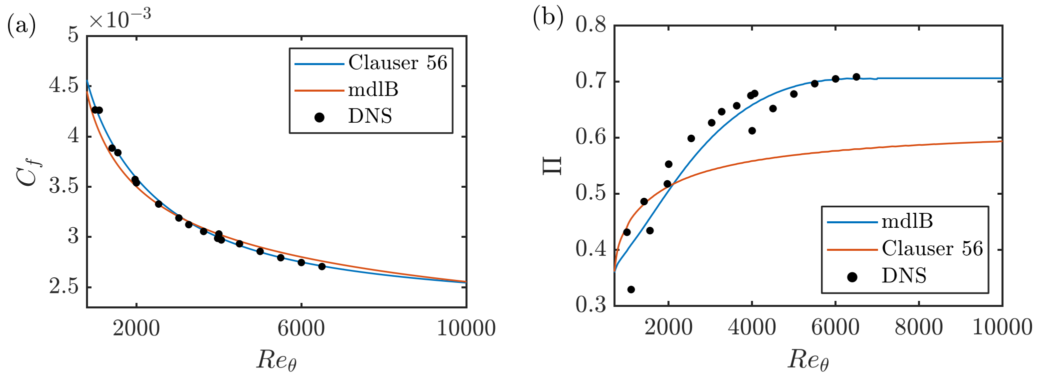

FIG. 6. (a) The skin friction coefficient C_f as a function of the Reynolds number Re_θ. (b) The wake parameter Π as a function of the Reynolds number Re_θ. We show mdlB, the empirical model in [18], and DNS.

We see that both Clauser's model [18] and mdlB agree with the DNS data and work well at high Reynolds numbers. In Fig. 6(b), mdlB follows roughly the DNS data. The Π value asymptotes to about 0.7 according to mdlB and about 0.6 according to Clauser [18].

D. Channel with small system rotation in an arbitrary direction

Both channel and boundary-layer flows have been extensively studied in the past, and we know, more or less, the answer to the two problems. In this subsection, we further build on mdlC and train a network for the eddy viscosity in a rotating channel. The rotation axis is in an arbitrary direction, but the rotation is small. That is, $\Omega_y^+ U^+ \lesssim O(0.1)$ and $\sqrt{\Omega_x^2+\Omega_z^2}^+ W^+ \lesssim O(0.1)$, where Ω_x, Ω_y, and Ω_z are the rotations in the x, y (wall-normal), and z directions and U, V, and W are the mean velocities in the x, y, and z directions. The reader is directed to [36] Sec. II A for more details of the flow.

The Reynolds averaged momentum equation in a fully developed channel with system rotation reads

$$\frac{\partial\langle u'v'\rangle}{\partial y} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x} + \nu\frac{d^2U}{dy^2} - \Omega_y W, \quad \frac{\partial\langle w'v'\rangle}{\partial y} = \nu\frac{d^2W}{dy^2} + \Omega_y U. \tag{13}$$

Here, $V=0$ because of continuity, U and W are functions of y only because the channel is fully developed, and $\langle p\rangle$ is a function of both x and y, hence the ∂ symbol instead of the d symbol. If one uses the viscous units to scale Eq. (13) and keep only the O(1) terms, the z direction momentum equation is trivial, and the x direction momentum equation becomes

$$\frac{\partial\langle u'v'\rangle}{\partial y} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x} + \nu\frac{d^2U}{dy^2}. \tag{14}$$

Hence, small system rotation does not fundamentally change the flow dynamics but only modifies the balance between the terms. Again, the reader is directed to [36] for more details. Invoking still the eddy viscosity, Eq. (14) becomes

$$0 = \frac{d}{dy}\left[(\nu_t+\nu)\frac{dU_{||}}{dy}\right] - \frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x}. \tag{15}$$

We build on the machine learning model mdlC and further account for the effect of system rotation. Specifically, we will train a bias-free neural network netR such that

$$\nu_t^+/y^+ = \mathrm{netL}(y^+) + \mathrm{netC}(y^+(y/h), y/h) + \mathrm{netR}(y^+(y/h)|\Omega|^+, y/h|\Omega|^+, \Omega_x^+, \Omega_y^+, \Omega_z^+), \tag{16}$$

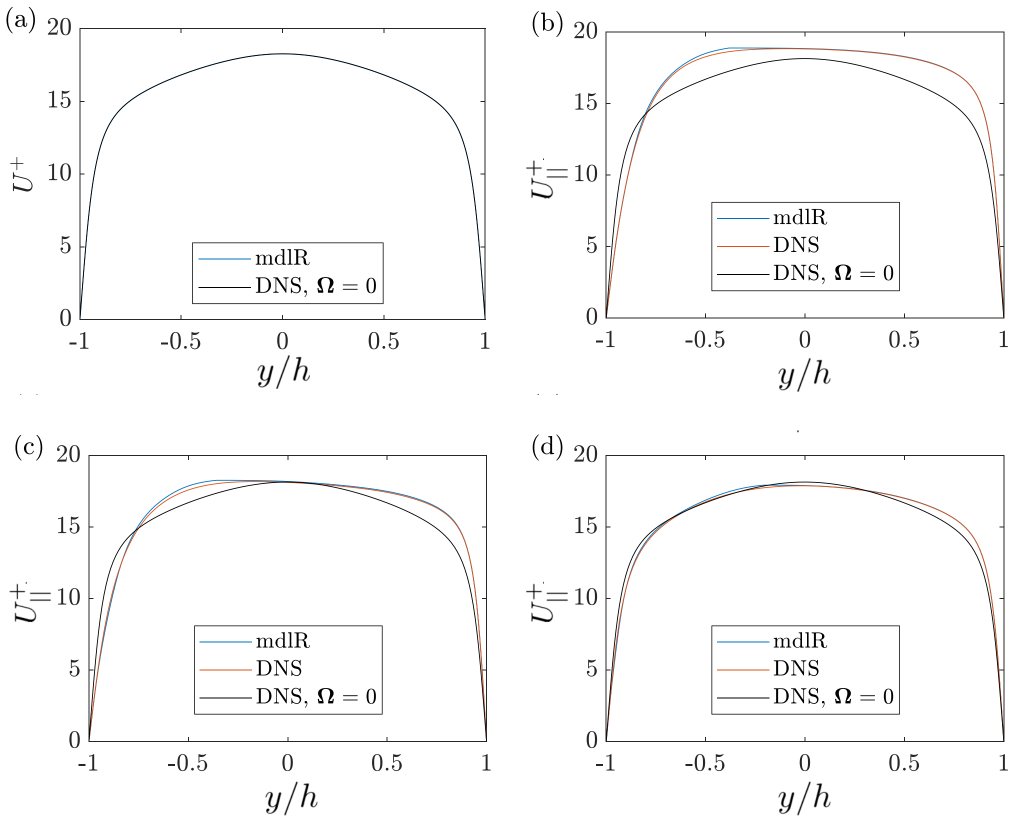

FIG. 7. *A posteriori* tests. Mean velocity as a function of the wall-normal coordinate in a rotating channel. Velocity normalization here is by the bulk friction velocity u_τ. The flow conditions are (a) $\Omega = (0,\ 0,\ 0)$, (b) $\Omega = (1,\ 0,\ -1)$, (c) $\Omega = (0,\ -0.01,\ -1)$, and (d) $\Omega = (-0.597,\ 0.006,\ -0.450)$.

where $\Omega = (\Omega_x,\ \Omega_y,\ \Omega_z)$. The inputs to netR are zero when $\Omega = 0$, and therefore netR gives zero when $\Omega = 0$ per the neutral neural network theorem. Consequently, mdlR fully respects mdlC. Moreover, since mdlC fully respects mdlL, mdlR fully respects mdlL and the law of the wall. The resulting model and all its results are referred to as mdlR.

The DNS data in [36] are used for training. The bulk friction Reynolds number is $\mathrm{Re}_\tau = u_\tau h/\nu = 180$. The system rotation is $-1 < \Omega_x^+ < 1$, $-0.01 < \Omega_y < 0.01$, and $-1 < \Omega_z < 1$. In [36], the authors surveyed the three-dimensional Ω space four times following a Bayesian method, leading to four datasets. We use three of these four datasets for training, i.e., S1, S2, and S3 per the notation in [36], and one dataset for testing, i.e., S4 per the notation in [36]. The training dataset contains 105 DNSs and the testing dataset contains 44 DNSs. The training data are preprocessed to avoid negative eddy viscosity (for numerical stability). The trained model is employed to solve Eq. (15) with the following boundary conditions:

$$\tau_{w,t}^+ + \tau_{w,b}^+ = 2, \tag{17}$$

where $\tau_{w,t}$ and $\tau_{w,b}$ are the top and the bottom wall shear stresses.

Figure 7 compares the mdlR to the DNSs in the test dataset at four conditions. We see from Fig. 7(a) that mdlR fully respects mdlC when $|\Omega| = 0$. In addition, we see from Figs. 7(b)–7(d) that mdlR is able to very accurately predict the mean flow at unseen conditions in the test dataset.

IV. CONCLUDING REMARKS

A paradigm for machine learning, namely, progressive machine learning, is proposed. It allows one to control the network's behavior when extrapolating and progressively evolve a simple model to more complex ones—both are long standing challenges in data-enabled turbulence modeling. In this process, the more complex models will always respect the less complex models, and training will not "violate" a previously working model.

We work through four examples to illustrate progressive machine learning: log layer, channel, boundary layer, and rotating channel. A neural network is trained for the eddy viscosity in these flows. Galilean invariance is guaranteed as we invoke the NS equation. The log layer physics is simple, and the resulting model mdlL is the model we evolve in the next three examples. We show that progressive learning leads to a data-enabled model that preserves the law of the wall and is open to corrections that account for other physics. In these examples, the physics that makes a more complex flow complex is absent in the less complex flow. This makes progressive machine learning straightforward. If the physics that makes a more complex flow complex is also present in the less complex flow, the present framework will not be able to separately train the two flows—although, in that case, the two flows may well be considered of similar complexities.

The topic of this paper is turbulence modeling. On its face, the technical problem is catastrophic forgetting [37]: a model forgets about a previous task when trained for a new task. Catastrophic forgetting arises when tasks are presented sequentially. It can be handled through continual learning [38] or by presenting the tasks altogether instead of sequentially. However, these methods will not help turbulence modeling. First, some flows are more fundamental than others for turbulence modeling, where tasks in continual learning do not have a hierarchy. Second, the network size does not change in continual learning, but progressive machine learning continuously expands the network size by including corrections. Third, continual learning usually does not emphasize extrapolation, but extrapolation to unseen Reynolds numbers is at the center of progressive machine learning [39].

ACKNOWLEDGMENTS

Y.B.'s visit at Pennsylvania State University is financially supported by National Natural Science Foundation of China Grant No. 91752202. L.C. acknowledges National Numerical Windtunnel Project No. 2018ZT1B01 for financial support. X.I.A.Y. acknowledges Office of Naval Research for financial support. X.I.A.Y. and G.H. thank P. Spalart for constructive comments.

[1] P. Spalart, Comments on the feasibility of LES for wings, and on hybrid RANS/LES approach: Advances in DNS/LES, in *Proceedings of the First AFOSR International Conference on DNS/LES*, 1997.

[2] H. Choi and P. Moin, Grid-point requirements for large eddy simulation: Chapman's estimates revisited, Phys. Fluids **24**, 011702 (2012).

[3] X. I. A. Yang and K. P. Griffin, Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation, Phys. Fluids **33**, 015108 (2021).

[4] C. Meneveau and J. Katz, Scale-invariance and turbulence models for large-eddy simulation, Annu. Rev. Fluid Mech. **32**, 1 (2000).

[5] S. T. Bose and G. I. Park, Wall-modeled large-eddy simulation for complex turbulent flows, Annu. Rev. Fluid Mech. **50**, 535 (2018).

[6] P. A. Durbin, Some recent developments in turbulence closure modeling, Annu. Rev. Fluid Mech. **50**, 77 (2018).

[7] X. I. A. Yang, J. Sadique, R. Mittal, and C. Meneveau, Integral wall model for large eddy simulations of wall-bounded turbulent flows, Phys. Fluids **27**, 025112 (2015).

[8] X. I. A. Yang, J. Sadique, R. Mittal, and C. Meneveau, Exponential roughness layer and analytical model for turbulent boundary layer flow over rectangular-prism roughness elements, J. Fluid Mech. **789**, 127 (2016).

[9] K. Duraisamy, G. Iaccarino, and H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. **51**, 357 (2019).

[10] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. **52**, 477 (2020).

[11] X. I. A. Yang, S. Zafar, J.-X. Wang, and H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids **4**, 034602 (2019).

[12] X. L. Huang, X. I. Yang, and R. F. Kunz, Wall-modeled large-eddy simulations of spanwise rotating turbulent channels-comparing a physics-based approach and a data-based approach, Phys. Fluids **31**, 125105 (2019).

[13] S. B. Pope and S. B. Pope, *Turbulent Flows* (Cambridge University, New York, 2000).

[14] C. Teo, K. Lim, G. Hong, and M. Yeo, A neural net approach in analyzing photograph in PIV, in *Conference Proceedings 1991: IEEE International Conference on Systems, Man, and Cybernetics* (IEEE, New York, 1991), pp. 1535–1538.

[15] M. Milano and P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. **182**, 1 (2002).

[16] L. Prandtl, Bericht über untersuchungen zur ausgebildeten turbulenz, Z. Angew. Math. Mech. **5**, 136 (1925).

[17] B. Launder and D. Spalding, The numerical computation of turbulent flows, Comput. Methods Appl. Mech. Eng. **3**, 269 (1974).

[18] F. H. Clauser, The turbulent boundary layer, in *Advances in Applied Mechanics* (Elsevier, Amsterdam, 1956), Vol. 4, pp. 1–51.

[19] X. I. A. Yang, Z.-H. Xia, J. Lee, Y. Lv, and J. Yuan, Mean flow scaling in a spanwise rotating channel, Phys. Rev. Fluids **5**, 074603 (2020).

[20] B. Tracey, K. Duraisamy, and J. Alonso, Application of supervised learning to quantify uncertainties in turbulence and combustion modeling, in *Proceedings of the 51st AIAA Aerospace Sciences Meeting* (AIAA, 2013), p. 259.

[21] J. Ling, R. Jones, and J. Templeton, Machine learning strategies for systems with invariance properties, J. Comput. Phys. **318**, 22 (2016).

[22] J. Weatheritt and R. Sandberg, A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship, J. Comput. Phys. **325**, 22 (2016).

[23] J.-L. Wu, H. Xiao, and E. Paterson, Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework, Phys. Rev. Fluids **3**, 074602 (2018).

[24] S. Beetham, R. O. Fox, and J. Capecelatro, Sparse identification of multiphase turbulence closures for coupled fluid-particle flows, J. Fluid Mech. **914**, A11 (2021).

[25] J. Sirignano, J. F. MacArt, and J. B. Freund, A deep learning PDE augmentation method with application to large-eddy simulation, J. Comput. Phys. **423**, 109811 (2020).

[26] J. R. Holland, J. D. Baeder, and K. Duraisamy, Field inversion and machine learning with embedded neural networks: Physics-consistent neural network training, in *AIAA Aviation*, p. 3200, 2019.

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

[28] T.-R. Xiang, X. Yang, and Y.-P. Shi, Neuroevolution-enabled adaptation of the Jacobi method for Poisson's equation with density discontinuities, Theoretical and Applied Mechanics Letters **11**, 100252 (2021).

[29] H. D. Akolekar, Y. Zhao, R. D. Sandberg, and R. Pacciani, Integration of machine learning and computational fluid dynamics to develop turbulence models for improved low-pressure turbine wake mixing prediction, Journal of Turbomachinery **143**, 121001 (2021).

[30] P. M. Milani, J. Ling, G. Saez-Mischlich, J. Bodart, and J. K. Eaton, A machine learning approach for determining the turbulent diffusivity in film cooling flows, Journal of Turbomachinery **140**, 021006 (2018).

[31] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. **807**, 155 (2016).

[32] P. Spalart, Comments on machine learning models, in *Model Consistent Data Driven Turbulence Modeling* (Symposium, Virtual, 2021).

[33] J. Graham, K. Kanov, X. I. A. Yang, M. Lee, N. Malaya, C. C. Lalescu, R. Burns, G. Eyink, A. Szalay, R. D. Moser, and C. Meneveau, A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for les, J. Turbul. **17**, 181 (2016).

[34] M. Lee and R. D. Moser, Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau \approx 5200$, J. Fluid Mech. **774**, 395 (2015).

[35] P. Schlatter and R. Örlü, Assessment of direct numerical simulation data of turbulent boundary layers, J. Fluid Mech. **659**, 116 (2010).

[36] X. L. Huang and X. I. Yang, A Bayesian approach to the mean flow in a channel with small but arbitrarily directional system rotation, Phys. Fluids **33**, 015103 (2021).

[37] R. M. French, Catastrophic forgetting in connectionist networks, Trends Cognit. Sci. **3**, 128 (1999).

[38] G. M. Van de Ven and A. S. Tolias, Three scenarios for continual learning, arXiv:1904.07734.

[39] C. Huang, S. Li, J. Wang, B. Yang, and F. Zhang, Global uncertainty analysis for the rrkm/master equation modeling of a typical multi-well and multi-channel reaction system, Combust. Flame **216**, 62 (2020).