MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™

PAPER • OPEN ACCESS

Machine-learned closure of URANS for stably stratified turbulence: connecting physical timescales & data hyperparameters of deep time-series models

IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 045063 https://doi.org/10.1088/2632-2153/ad9704

MACHINE LEARNING Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
10 July 2024

REVISED
23 October 2024

ACCEPTED FOR PUBLICATION
25 November 2024

PUBLISHED
12 December 2024

PAPER

Machine-learned closure of URANS for stably stratified turbulence: connecting physical timescales & data hyperparameters of deep time-series models

Muralikrishnan Gopalakrishnan Meena[1,*], Demetri Liousas[2,4], Andrew D Simin[2,5], Aditya Kashi[1], Wesley H Brewer[1], James J Riley[3] and Stephen M de Bruyn Kops[2]

[1] National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831, United States of America
[2] Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Amherst, MA 01003, United States of America
[3] Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, United States of America
[4] Present Address: MIT Lincoln Laboratory, Lexington, MA 02421, United States of America.
[5] Present Address: DoD HPCMP PET/GDIT, Vicksburg, MS 39180, United States of America.
[*] Author to whom any correspondence should be addressed.

E-mail: gopalakrishm@ornl.gov

Keywords: unsteady RANS, stably stratified turbulence, deep time-series models, timescale extraction

Abstract

Stably stratified turbulence (SST), a model that is representative of the turbulence found in the oceans and atmosphere, is strongly affected by fine balances between forces and becomes more anisotropic in time for decaying scenarios. Moreover, there is a limited understanding of the physical phenomena described by some of the terms in the Unsteady Reynolds-Averaged Navier–Stokes (URANS) equations—used to numerically simulate approximate solutions for such turbulent flows. Rather than attempting to model each term in URANS separately, it is attractive to explore the capability of machine learning (ML) to model groups of terms, i.e. to directly model the force balances. We develop deep time-series ML for closure modeling of the URANS equations applied to SST. We consider decaying SST which are homogeneous and stably stratified by a uniform density gradient, enabling dimensionality reduction. We consider two time-series ML models: long short-term memory and neural ordinary differential equation. Both models perform accurately and are numerically stable in *a posteriori* (online) tests. Furthermore, we explore the data requirements of the time-series ML models by extracting physically relevant timescales of the complex system. We find that the ratio of the timescales of the minimum information required by the ML models to accurately capture the dynamics of the SST corresponds to the Reynolds number of the flow. The current framework provides the backbone to explore the capability of such models to capture the dynamics of high-dimensional complex dynamical system like SST flows[6].

1. Introduction

Stably stratified turbulence (SST) is a model for understanding fluid flows that involve an ambient stable density stratification in the oceans due to temperature or salt gradients. Studying SST has important implications for climate modeling, pollution mitigation, and deep sea mining. In general the dynamics of SST reflect a competition between many forces including inertial, viscous, buoyancy, Coriolis, and shearing. This interplay between the various forces and length scales is directly reflected on the scaling of energy and

[6] **Notice:** This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

buoyancy spectra, which change based on the strength of stratification [1–3]. Even in the simplest case, though, SST can be parameterized by a minimum of two dimensionless groups defined by ratios of the inertial, buoyancy, and viscous forces. These ratios can be chosen in a variety of ways, and we use Reynolds and Froude numbers defined later. Consistent with the parameter space of SST being, at a minimum, two dimensional, the flows are highly intermittent and anisotropic at large scales. Riley and Lindborg [4] provide an overview of SST, de Bruyn Kops and Riley [5] review many theoretical, laboratory, and numerical studies in SST, and Mater and Venayagamoorthy [6] present a framework for parameterizing a three parameter variant of SST that includes mean shear in order to describe the flow dynamics in terms of where the flow resides in parameter space.

The extreme scales required to capture the evolution of such flows in nature hinder computational experiments even with the latest high-performance computing platforms [7] and thus, the atmospheric and oceanographic communities have relied on parametrizations or surrogate models to capture the effect of the small scale flow features in SST. Turbulence closure models in unsteady Reynolds averaged Navier–Stokes (URANS) simulations and large-eddy simulations (LES) of the atmosphere, ocean, and their sub-systems are broad classes among many such turbulence models. Particularly, URANS second-moment closure (SMC) based modeling has been prevalent in simulations of stratified shear layers and wakes [8, 9]. Such models are limited by the appropriate choice of the parametrization, interpretation of the parametrizations, and generalizability with different flow regimes and types. Data-driven techniques are attractive to overcome such challenges. Over the past decade, there is a history of using machine learning (ML) methods to tackle fluid flow problems [10], and specifically, to assist in the augmentation of turbulence models [11]. The atmospheric and oceanographic communities have been using ML to formulate surrogate models to capture the effects of sub-grid scale structures. Such efforts have been focused on turbulence closures for LES and global climate models [12–19]. There have been various efforts to use data-driven models in modeling other applications of stratified turbulent flows as well, such as shear layers [20], stratified wakes [21], engineering applications [22–24], optical turbulence [25], and mixing in SST [26–30].

Developing predictive capability for SST is the first motivation for this study. A second is to apply ML to the difficult problem of closing the URANS equations or the related approach of LES. In URANS, differential equations for averaged quantities are solved numerically with models for terms involving fluctuations about the averages. In LES, equations are solved for quantities resolved on the numerical grid and subgrid-scale quantities are modeled. With URANS, even in configurations involving just two types of forces, e.g. inertial and viscous, all but the simplest closure models typically involve differences between model terms. SST flows involve more than two forces, so there are more terms to be modeled in either URANS or LES, and there is limited understanding of the physical phenomena described by some of these terms. Rather than attempting to model each term separately, as is done in SMC modelling, it is attractive to see how well ML can be used to model groups of terms, that is, to directly model the force balances.

Furthermore, the inherent nonlinear dynamics of the SST, even in a reduced-order sense, pose the challenge of finding interpretable and generalized surrogates. Thus, the third and final motivation of this study is to formulate interpretable ML models that accurately predict force balances. Specifically, we will focus on interpreting the data requirements of time-series-based ML models for modeling a reduced dimensional representation of canonical SST.

Advances in ML have shown the capability of ML models to implicitly learn the underlying dynamics of a time-dependent physical system [31–34]. Recurrent neural networks, such as long short-term memory (LSTM) [35–37] and gated recurrent units (GRUs) [38, 39], have been shown to model and predict time-series data effectively. Moreover, recent works have shown effective utilization of time-series ML models for modeling turbulent flow data [40, 41]. Thus, time-series ML models could provide a computationally inexpensive way to simulate complex time evolving force balance of the SST URANS equations. The task of interpreting such models still remains a challenge [42]. State-of-the-art approaches involve using techniques such as model visualization, layer-wise relevance propagation, and attention mechanisms [43–45]. While such techniques provide valuable insights into the working of the model, extraction of physical knowledge about the system being modeled is limited. One step in this direction is to use ML models for extracting the timescales of the physical system, which are fundamental to understanding complex dynamical systems.

In this paper, we report on the use of time-series ML to close the URANS equations for one of the simplest configurations of SST, namely, homogeneous and decaying turbulence. The SST configuration and URANS equations are presented in section 2. In section 3, we review time-series ML approaches, the training data sets, and techniques for interpreting the data requirements of the ML models. The results from our URANS models are discussed in section 4. Some conclusions are drawn in section 5.

2. Theoretical background for URANS of SST

The measure of importance of stratification in SST is generally quantified using the nondimensional Froude number, Fr, which is the ratio of the inertial and buoyancy forces. The flow is highly stratified and anisotropic when $\mathrm{Fr} \ll 1$ and weakly stratified when $\mathrm{Fr} \geqslant 1$ [46, 47]. For this work, we consider temporally evolving SST with low Fr, that is axisymmetric in the sense that the statistics are independent of horizontal direction. The flow is stably stratified by a uniform density gradient so that it is homogeneous in the vertical direction. While the configuration is formally homogeneous in all directions, we show in section 3.3 with contour plots from direct numerical simulations (DNS) that the flow is intermittent at length scales that are a significant fraction of the domain size. We take advantage of the homogeneity to reduce the RANS grid down to one point in each direction in section 3.3. Here, let us write the governing equations and their Reynolds-averaged versions for the general case.

The modeled flow evolves with density stratification and is governed by the dimensionless Navier–Stokes equations satisfying the Boussinesq assumption, given by

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\left(\frac{2\pi}{\mathrm{Fr}}\right)^2 \rho \boldsymbol{e}_z - \nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u}, \tag{2}$$

$$\frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho - w = \frac{1}{\mathrm{RePr}} \nabla^2 \rho. \tag{3}$$

Here, $\boldsymbol{u} = (u, v, w)$ represents the velocity vector in the Cartesian coordinate system $\boldsymbol{x} = (x, y, z)$ evolving in time t, and ρ and p are the deviations of density and pressure from their ambient conditions, respectively. The velocity and length scales for non-dimensionalzing the variables are the r.m.s. velocity $\hat{U}$ and integral length scale $\hat{L}$, respectively. Note that the dimensional quantities are represented with $\hat{\cdot}$. Time t is non-dimensionalized as $t = (\hat{t} - \hat{t}_0)/\hat{t}_{\mathrm{LE}}$, where $\hat{t}_0$ is the initial time at which the turbulence decay process starts and $\hat{t}_{\mathrm{LE}}$ is the large-eddy advective time scale. The non-dimensional quantities, Froude number Fr, Reynolds number Re, and Prandtl number Pr, are defined respectively as

$$\mathrm{Fr} = \frac{2\pi \hat{U}}{\hat{N}\hat{L}}, \mathrm{Re} = \frac{\hat{U}\hat{L}}{\hat{\nu}}, \text{ and } \mathrm{Pr} = \frac{\hat{\nu}}{\hat{\alpha}}. \tag{4}$$

Here, $\hat{N} = \left[-(\hat{g}/\hat{\rho}_0)(\mathrm{d}\overline{\hat{\rho}}/\mathrm{d}\hat{z})\right]^{1/2}$ is the Brunt–Väisälä frequency or the buoyancy frequency, $\hat{g}$ is the acceleration due to gravity in the $-z$ direction, $\hat{\rho}_0$ is the density at a reference height of $\hat{z}_0$, $\hat{\nu}$ is the kinematic viscosity, and $\hat{\alpha}$ is the thermal diffusivity. The density gradient $\mathrm{d}\overline{\hat{\rho}}/\mathrm{d}\hat{z}$ is a uniform, time-invariant, stable ambient density gradient applied to impose stratification.

For the homogeneous case, we can approximate the Reynolds average with a spatial average over the entire domain, denoted by $\langle \cdot \rangle$. Then the time evolution of the average flow can be described in terms of energy equations by decomposing kinetic energy into its horizontal and vertical contributions, E_{H} and E_{V}, which are coupled via pressure. The potential energy E_{P} is a scalar multiple of the variance of the buoyancy. An additional equation is required for the buoyancy flux, which couples E_{V} and E_{P}. The resulting set of evolution equations is

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle E_{\mathrm{H}}\rangle = \underbrace{\langle p \nabla_{\mathrm{H}} \cdot \boldsymbol{u}_{\mathrm{H}}\rangle}_{\text{needs model}} - \hat{\nu} \underbrace{\left\langle \left(\frac{\partial \boldsymbol{u}_{\mathrm{H}}}{\partial x_j}\right)^2 \right\rangle}_{\text{needs model}} \tag{5}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle E_{\mathrm{V}}\rangle = \underbrace{\left\langle p \frac{\partial w}{\partial z}\right\rangle}_{\text{needs model}} - \langle bw\rangle - \hat{\nu} \underbrace{\left\langle \left(\frac{\partial w}{\partial x_j}\right)^2 \right\rangle}_{\text{needs model}} \tag{6}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle E_{\mathrm{P}}\rangle = \langle bw\rangle - \underbrace{\frac{\hat{\alpha}}{\hat{N}^2} \left\langle \left(\frac{\partial b}{\partial x_j}\right)^2 \right\rangle}_{\text{needs model}} \tag{7}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle bw\rangle = \underbrace{\left\langle p \frac{\partial b}{\partial z}\right\rangle}_{\text{needs model}} + \langle w^2\rangle \hat{N}^2 - \langle b^2\rangle - \underbrace{(\hat{\nu} + \hat{\alpha}) \left\langle \left(\frac{\partial b}{\partial x_j}\right)\left(\frac{\partial w}{\partial x_j}\right) \right\rangle}_{\text{needs model}} \tag{8}$$

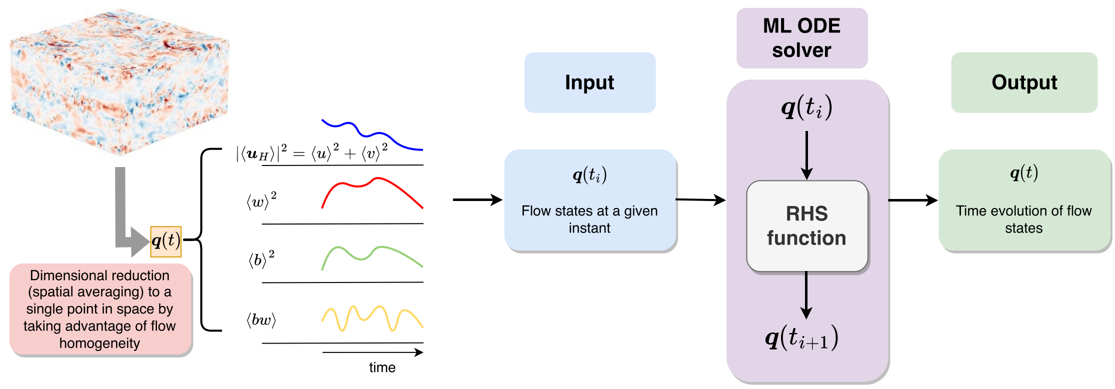

Figure 1. Framework of the current modeling approach to predict the time evolution of the reduced-dimensional flow state variables.

where $E_\mathrm{H} = (1/2)|\boldsymbol{u}|_\mathrm{H}^2$, $E_\mathrm{V} = (1/2)w^2$, and $E_\mathrm{P} = b^2/(2\hat{N}^2)$. Here, $b = \frac{\hat{g}}{\hat{\rho}_0}\hat{\rho}$ is the buoyancy, and $|\boldsymbol{u}_\mathrm{H}| = \left(u^2+v^2\right)^{1/2}$ is the horizontal velocity. Several terms on the right-hand-sides (RHSs) of these equations are marked to indicate that they must be modeled in order to close the system of ordinary differential equations. Rather than model each term, or even the RHS of each equation, our objective is to model the entire set of RHSs using ML techniques, as broadly depicted in figure 1.

3. Approach

3.1. Procedure

As mentioned in section 2, the objective is to model the temporal evolution of the energies (horizontal and vertical kinetic energies, and potential energy) and buoyancy flux for a single point in space. Approximating the derivatives of the energy variables are of interest for modeling the RHS of equations (5)–(8), needed to close the system. The terms to be modeled can be represented as

$$\boldsymbol{f} = \left[\frac{\mathrm{d}}{\mathrm{d}t}\langle E_\mathrm{H}\rangle, \quad \frac{\mathrm{d}}{\mathrm{d}t}\langle E_\mathrm{V}\rangle, \quad \frac{\mathrm{d}}{\mathrm{d}t}\langle E_\mathrm{P}\rangle, \quad \frac{\mathrm{d}}{\mathrm{d}t}\langle bw\rangle\right]^T. \tag{9}$$

The terms are non-dimensionalized by the total energy, $E_\mathrm{T} = \langle E_\mathrm{H}\rangle + \langle E_\mathrm{V}\rangle + \langle E_\mathrm{P}\rangle$, and buoyancy frequency, $\hat{N}$, at the initial condition, giving $\langle E_\mathrm{H}\rangle = (1/2)|\langle \boldsymbol{u}_\mathrm{H}\rangle|^2/E_{\mathrm{T}_0}$, $\langle E_\mathrm{V}\rangle = (1/2)\langle w\rangle^2/E_{\mathrm{T}_0}$, $\langle E_\mathrm{P}\rangle = \langle b\rangle^2/(2\hat{N}^2 E_{\mathrm{T}_0})$, and $\langle bw\rangle/(2\hat{N}E_{\mathrm{T}_0})$ as the terms to be modeled. Here, we can represent the independent variables as $\boldsymbol{q} = [|\langle \boldsymbol{u}_\mathrm{H}\rangle|^2, \quad \langle w\rangle^2, \quad \langle b\rangle^2, \quad \langle bw\rangle]^T$. Thus, the problem can be generically represented in discrete time as

$$\frac{\mathrm{d}\boldsymbol{q}(t_i)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{q}(t_i), t_i) \tag{10}$$

where t_i represents a discrete time instant. We use supervised deep learning techniques [48] to model the relationship between $\boldsymbol{q}$ and $\boldsymbol{f}$, and thus, model the time evolution of $\boldsymbol{q}$, as shown in figure 1.

Neural networks have the capacity to approximate any Borel measurable function if it has the proper architecture [49]. We are interested in neural networks capable of modeling the RHS of the system of ODEs in equation (10), representing the reduced form of equations (5)–(8), so that any numerical ODE solver can solve the time evolution of the states, $\boldsymbol{q}(t)$. Moreover, as the problem is fundamentally dependent on time and the time history of the states, modeling the memory effect present in the time series using time-series ML models is attractive. Thus, our objective revolves around using sequences of time-series data to predict the time evolution of $\boldsymbol{q}$. Here, sequence refers to the ordered snapshots of state time history (or memory) used to model the dynamics of the system.

Generally, sequences of time-series data have two parameters analogous to timescales in physical time-evolving systems, as illustrated in figure 2. One timescale is the sequence length, which is the time span of the data in a single sequence. The second timescale is the sampling period of the data within one sequence. We will refer to latter timescale by its inverse, the sampling frequency. With respect to physical timescales, the sequence length can be considered as the largest timescale of the system being modeled whereas the sampling period can be treated as the shortest timescale of the system. We will elaborate more on what these timescale mean for SST in section 4.

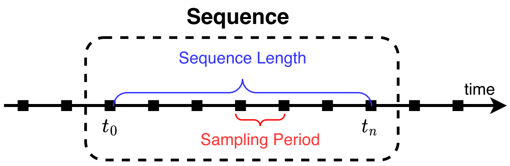

Figure 2. Illustration of the timescales defined by a sequence in a time-series data.

We use these underlying timescales of the temporal data as hyperparameters of time-series ML models. Furthermore, we determine the minimum information, based on these timescales, required by the ML models to effectively model the system dynamics. These techniques are used specifically for interpreting the data requirements for the time-series ML models.

We choose to model the evolution of the flow states using two routes: (1) model the time derivatives (or the entire RHS of equations (5)–(8) using an ML model and pass this information into an ODE solver to obtain the time evolution of $\boldsymbol{q}$ and (2) directly model the time evolution of $\boldsymbol{q}$ using an ML framework. We achieve these two routes using the LSTM [35] and neural ordinary differential equation (NODE) [50] models, respectively. In the following sections, we describe the architecture of the LSTM and NODE models, and the dataset used to train these models.

3.2. Neural network (NN) setup

Without a loss of generality, the system of differential equations governing the SST, equation (10), are modeled as a NN, $\mathcal{N}$, given by

$$\frac{\mathrm{d}\boldsymbol{q}(t_i)}{\mathrm{d}t} = \underbrace{\tilde{\boldsymbol{f}}(\boldsymbol{q}(t_i), t_i, \theta)}_{\mathcal{N}} \tag{11}$$

where the NN is parameterized by a tensor, θ, containing learnable weights and biases defining the map between the input and output layers of the network. The symbol $\tilde{\cdot}$ represents the quantities obtained from the learned ML model.

3.2.1. LSTM model

In the first route for modeling the system of ODEs, given the components of $\boldsymbol{q}$ at an instant in time t_i, we use NNs of the form $\mathcal{N}(\boldsymbol{q}, \theta)$ to model the RHS of equation (11), with θ representing the parameters of the NN ('weights'). Specifically, we use the LSTM [35] network as the NN model. An LSTM architecture is a sub-class of the recurrent NN (RNN) architecture, which uses internal states of nodes (memory) to process arbitrary sequences of inputs and thus, allows for time-series to be modeled [31, 33, 34]. LSTM is commonly used for time-series prediction because of its improved stability and predictive capabilities over generic RNNs [35]. A sample depiction of the model is shown in figure 3.

The input to the LSTM model is the time history $\boldsymbol{q}(t_{i-n})$, $\boldsymbol{q}(t_{i-(n-1)})$, …, $\boldsymbol{q}(t_i)$ where n is the sequence length. The output of LSTM is $\tilde{\boldsymbol{f}}(\boldsymbol{q}(t_i), t_i, \theta)$, which is fed into an ODE solver (like a Runge–Kutta scheme) to obtain the next value in the series, $\boldsymbol{q}(t_{i+1})$. The process is repeated to predict the entire time evolution of $\boldsymbol{q}$. Thus, the entire workflow can be viewed as a LSTM+ODE numerical simulation, where the LSTM serves as the RHS of the ODE solver. The LSTM model is trained using the loss function comprised of the RHS of equation (11), given by

$$\mathcal{L}\left[\boldsymbol{f}(\boldsymbol{q}(t_i), t_i), \tilde{\boldsymbol{f}}(\boldsymbol{q}(t_i), t_i, \theta)\right]. \tag{12}$$

Hyperparameter selection for the LSTM model is done through experimentation (broad grid search). We found that for modeling the SST problem, the best predictions are obtained with four LSTM layers with a hidden size of 10. We also use a multi-layer perceptron (MLP) consisting of one hidden layer with 15 neurons before the output layer. A leaky rectified linear unit activation function with a slope of 0.1 is applied for both the LSTM and MLP layers. We measure the mean squared error as the loss function, $\mathcal{L}$, for optimizing the network weights during the training process. We select a learning rate of 0.001 and train the model for 4000 epochs, after which the model training converges.

3.2.2. NODE model

In the second route for modeling the system of ODEs, given the components of $\boldsymbol{q}$ at an instant in time t_i, we use an ML framework devised by Chen *et al* [50], NODE, to directly predict the time evolution of the system.

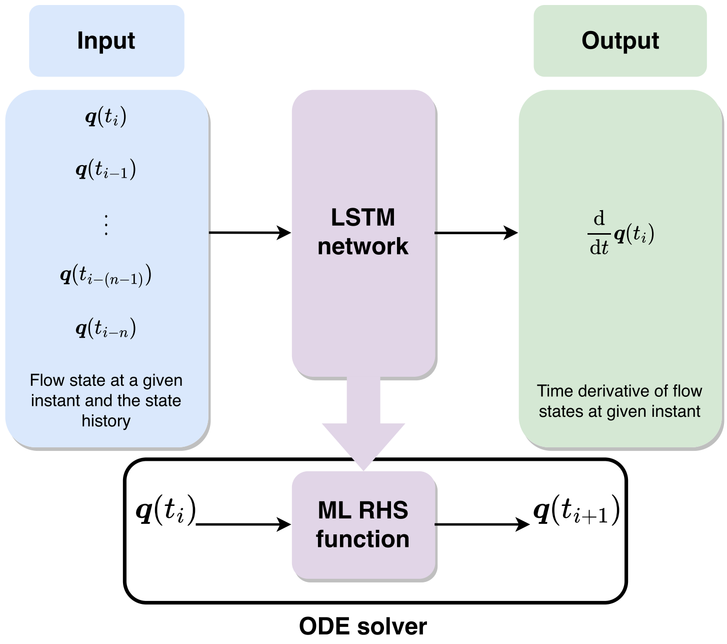

Figure 3. Sample portrayal of the LSTM modeling framework used in the current study for predicting the time evolution of a system of ODEs.

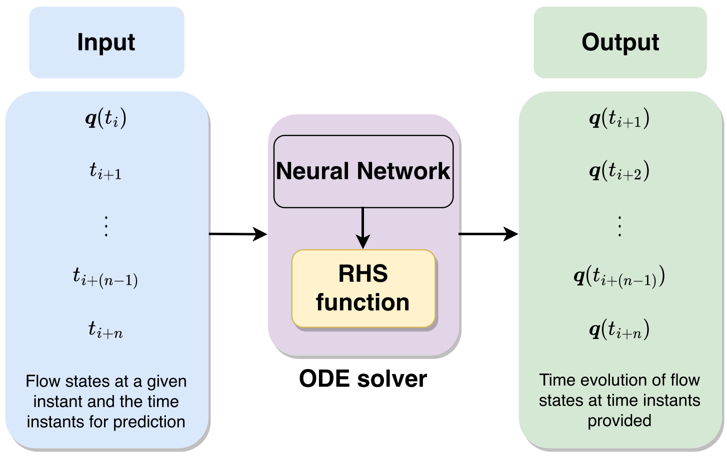

Figure 4. Sample portrayal of the NODE modeling framework used in the current study for predicting the time evolution of a system of ODEs.

The NODE framework models an arbitrary system of ODEs as an NN given discrete observations. Legaard *et al* [51] refer to this framework as a time-stepper model compared to direct solution models which map a given time, t_i, to its observation, $\boldsymbol{q}_i$, such as deep NNs (DNNs), RNNs, and residual neural networks (ResNet) [52]. Motivated by the similarity between ResNets and Euler's method for numerical integration, wrapping a black-box ODE solver around an NN offers many modeling advantages. The main advantage being that, unlike ResNets or traditional sequential learning architectures, transformations between hidden states evolve continuously. Many systems described by differential equations evolve as such, so the combination of well known mathematical integration tools and NNs can be a powerful paradigm in scientific applications. In recent years, NODE has been used to model complex systems in fluid dynamics, such as to model and predict thermoacoustic instability [53] and learn subgrid-scale models [54]. Related work also includes the implementation of ODE-based ML solutions for modeling dynamical systems [55], reduced-order modeling [56], and turbulent flow control [57, 58].

As depicted in figure 4, NODE takes as input the current state of the system $\boldsymbol{q}(t_i)$ and the desired time instants $t_{i+1}, t_{i+2}, \ldots, t_{i+n}$ at which to evaluate the state. The output is the state variables $\boldsymbol{q}(t_{i+1}), \boldsymbol{q}(t_{i+2}), \ldots,$ $\boldsymbol{q}(t_{i+n})$ at those discrete time instants. What makes a time-stepper model unique compared to direct

Table 1. Inputs and outputs of the LSTM and NODE models.

	LSTM	NODE
Inputs	$\boldsymbol{q}(t_{i-n}), \boldsymbol{q}(t_{i-(n-1)}), \ldots, \boldsymbol{q}(t_i)$	$\boldsymbol{q}(t_i)$, t_i, t_{i+1}, … , t_{i+n}
Outputs	$\boldsymbol{f}(t_i)$	$\boldsymbol{q}(t_{i+1})$, $\boldsymbol{q}(t_{i+2})$, …, $\boldsymbol{q}(t_{i+n})$

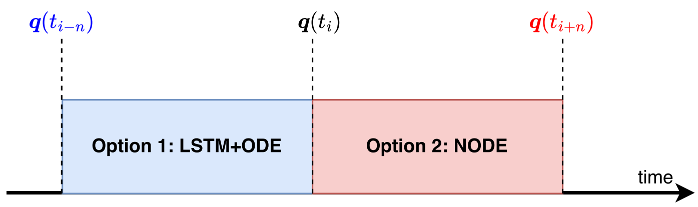

Figure 5. Comparing the time-dependence for the LSTM and NODE models.

solutions, is that the loss is computed after integrating the learned approximation of the differential equation, equation (11). As a result, the input to the scalar loss function $\mathcal{L}$ is the output of the ODE solver, given by

$$\begin{aligned}\mathcal{L}\left[\boldsymbol{q}\left(t_{i+n}\right), \tilde{\boldsymbol{q}}\left(t_{i+n}\right)\right] &= \mathcal{L}\left[\boldsymbol{q}\left(t_{i+n}\right), \boldsymbol{q}\left(t_i\right) + \int_{t_i}^{t_{i+n}} \tilde{\boldsymbol{f}}\left(\boldsymbol{q}\left(t\right), t, \theta\right) \mathrm{d}t\right] \\ &= \mathcal{L}\left[\boldsymbol{q}\left(t_{i+n}\right), \mathrm{ODESolve}\left(\boldsymbol{q}\left(t_i\right), \tilde{\boldsymbol{f}}, t_i, t_{i+n}, \theta\right)\right] \end{aligned} \tag{13}$$

where t_i and t_{i+n} represents the initial and final time instants for a given sample of the time-series data. Gradients can be computed using the adjoint sensitivity method, solving an augmented ODE backwards in time. The loss function can then be minimized by comparing the integrated prediction, $\tilde{\boldsymbol{q}}(t)$, and observed solution, $\boldsymbol{q}(t)$, in relation to the tensor, θ, which parameterizes the differential equation. Using this framework, recovering both the approximation of the state variables and their derivatives are convenient. Derivatives are recovered by inputting the predictions back into the NN defining the temporal derivative of the system, $\tilde{\boldsymbol{f}}(\boldsymbol{q}(t), t, \theta)$ (equation (11)).

In literature, NODE has been shown to be less sensitive to hyperparameter selection compared to traditional NN models, like MLPs and ResNets [59]. This pertains to the hyperparameters of the model architecture and not the two data hyperparameters of interest in the current study—sequence length and sampling frequency. For the model hyperparameters of the NODE architecture, we chose an absolute tolerance of 10^{-3} and a relative tolerance of 10^{-4} for the integrated ODE solver [50]. Furthermore, the MLP within the NODE model has 10 layers with 40 neurons per layer. We use a Sigmoid Linear Unit activation function for these layers. Mean squared error is used as the loss function, $\mathcal{L}$, for optimizing the network parameters, θ, during the training process. For this configuration, we used a higher learning rate of 0.05 compared to LSTM and lower number of epochs of 1000 to train the NODE model. This is because the NODE model converged faster with similar performance.

3.2.3. Comparing LSTM and NODE models

As described above, the time-series prediction that we formulate the LSTM and NODE models to perform are fundamentally different. In general, the inputs and outputs of the LSTM and NODE models can be summarized as shown in table 1 and their time-dependence can be summarized as shown in figure 5. While the LSTM model inherently relies on the time history of the data, the NODE model utilizes the future time evolution of the data to model the behaviour of the system. Nonetheless, there are two timescales for the input data to both the models—the sequence length and sampling frequency. Using these two timescales as the main hyperparameter of the two models, we investigate the minimum information (with respect to these two timescales) that is required to effectively model a system of ODEs—assessed in terms of the accuracy and numerical stability of the resulting ODE solver.

3.3. Training the models

The time-series models are trained using data from a decaying homogeneous SST flow simulation described in Riley and de Bruyn Kops [60]. A sample portrayal of the time evolution of the SST flow is shown in figure 6. The flow is initialized using Taylor–Green vortices with low-level noise (10% of initial energy) to break the symmetries of the vortices and trigger instabilities for the flow to evolve into turbulence. The

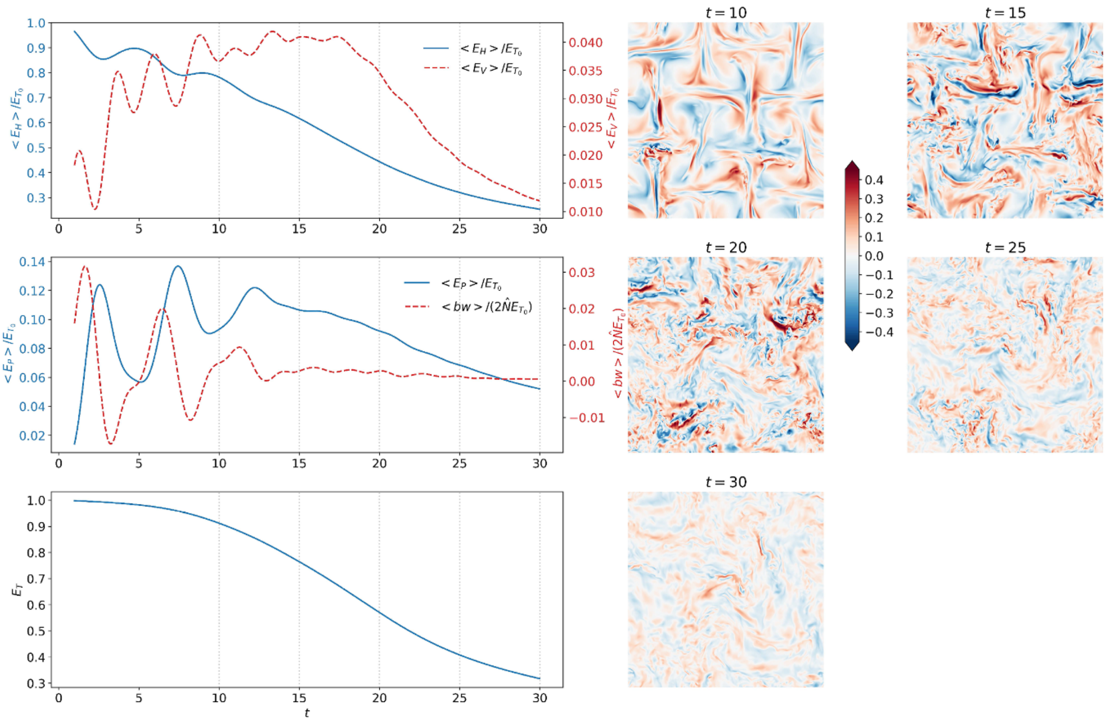

Figure 6. Time series data of spatially averaged non-dimensionalized energies and buoyancy flux (equations (5)–(8)) for the SST flow being modeled. Candidate horizontal slices of the flow field visualized using vertical velocity, w, at various time instants are also provided to depict the evolution of the flow through various flow regimes.

ambient stratification is constant and the initial density perturbation field is zero. This configuration is a significant test for URANS because it starts out laminar, then transitions to turbulence, and finally becomes increasingly more strongly stratified as it decays such that the inertial force decreases relative to the buoyancy force. We consider the case with initial Reynolds and Froude numbers $\mathrm{Re} = 3200$ and $\mathrm{Fr} = 4$, which is discussed in detail in [61, 62].

As mentioned in section 2, we take advantage of the homogeneity of the flow and spatially average the spatio-temporal data to a single grid point and obtain time series of the state variables. The training data consist of sequence of these time-series data, defined by two time scales: (1) the length of the sequence, sequence length, and (2) the time interval between samples within the sequence, the sampling frequency. We use these two variables as hyperparameters to interpret the data requirements of the models, where each model is trained for a single type of sequence defined by the hyperparameters. Note that these data hyperparameters are different from the model hyperparameters, which are chosen as mentioned in section 3.2 and kept constant throughout the analysis. For each hyperparameter set, the time-series is interpolated to have the same time-step as that of the sampling frequency. Thus, the number of samples used for training the models is determined by the sequence length as well as the sampling frequency.

For a given set of samples defined by the data hyperparameters, we split the time-series to training and testing regimes based on the temporal evolution of the system. We chose to train the models until $t = 20$, as the data captures both the transient as well as the beginning of the turbulent decay regimes. The testing is done both for the unseen data as well as from the beginning (in the training regime) to test both the accuracy and numerical stability of the model. The full training data for a given set of hyperparameters is randomly split in a 90 : 10 ratio for training and validation at each epoch, ensuring the model does not train on continuous segments of the dataset. As the terms modeled are already non-dimensionalized, we did not have to perform additional scaling of the data, which is typical for NN training due to the limited range in which activation functions can operate.

4. Results and discussion

Here, we evaluate the ability of time-series ML to learn the dynamics of the reduced-dimensional SST by modeling the system of equations governing the force balance of SST URANS equations. We demonstrate that data requirements of these models are interpretable by extracting the physical timescales corresponding to the information learned by the model. We study the relationship between the physical timescales and the

ML hyperparameters analogous to timescales of the time-series data—the data hyperparameters sequence length and sampling frequency.

To demonstrate the interpretability framework, we first model the time evolution of a damped pendulum [63] as a sample model problem. The damped pendulum is a very simple and well-studied nonlinear dynamical system. It has two timescales that can be derived analytically, which makes it a suitable choice for this study. Although the damped pendulum problem is not as complex as the nonlinear SST that motivates this study, we use the damped pendulum problem as a canonical model problem to demonstrate our framework of extracting physical timescales from time-series ML models [64]. Herein, we first show the results for the data hyperparameters (sequence length and sampling frequency) for which we obtained the best results and then we discuss the effects of these hyperparameters on the model performance and how they can be used to interpret the data requirements of the models.

4.1. Sample test problem: simple pendulum

ML models for both the LSTM and NODE are tested and validated with a simpler, well studied, synthetic dataset describing the motion of a damped pendulum. A damped pendulum exhibits some of the fundamental behaviours observed in the results for homogeneous SST, making it a relevant dataset for evaluating model performance. The governing equations of the pendulum are typically written in terms of its angular position θ and angular velocity ω. However, many dynamical systems in engineering and nature can be described by energy equations. In particular, the motivation for this study, URANS modeling of SST, is formulated in terms of energies as described in equations (5)–(8). The system of ODEs governing the time evolution of the pendulum in terms of its kinetic energy ($E_\mathrm{K} = \frac{1}{2}m\ell^2\omega^2$), potential energy ($E_\mathrm{P} = mg\ell(1-\cos\theta)$), and the flux ($F = mg\ell\omega\sin\theta$) between them is given by

$$\frac{\mathrm{d}E_\mathrm{K}}{\mathrm{d}t} = -\frac{2b}{m}E_\mathrm{K} - F \tag{14}$$

$$\frac{\mathrm{d}E_\mathrm{P}}{\mathrm{d}t} = F \tag{15}$$

$$\frac{\mathrm{d}F}{\mathrm{d}t} = 2\frac{g}{\ell}(E_\mathrm{K} - E_\mathrm{P}) + \frac{E_\mathrm{P}}{m\ell^2}(2E_\mathrm{K} - E_\mathrm{P}) - \frac{b}{m}F, \tag{16}$$

where the parameters of the pendulum are the damping coefficient b, mass m, acceleration due to gravity g, and length ℓ. Thus, the state of the system is defined as $\boldsymbol{q} = [E_\mathrm{K} \quad E_\mathrm{P} \quad F]^T$. Solving the system of ODEs requires the initial conditions E_{K_0}, E_{P_0}, and F_0. The similarities of the states of the pendulum and the SST problem are notable.

The pendulum has two fundamental timescales that govern its motion. They are the oscillating timescale τ_o and the damping timescale τ_d. These can be derived analytically using the small-angle approximation [63] and are given by

$$\tau_\mathrm{o} = \sqrt{\frac{\ell}{g}} \tag{17}$$

$$\tau_\mathrm{d} = \frac{m}{b}. \tag{18}$$

We can use these timescales to determine the theoretical minimum information that a data-driven model would need to effectively learn the dynamics of a pendulum. An accurate model must capture the information corresponding to both the damping as well as oscillating timescales.

4.1.1. LSTM and NODE results

We select a damped pendulum with damping coefficient $b = 0.5$, mass $m = 1$, acceleration due to gravity $g = 25$, length $\ell = l$, and initial conditions $E_{\mathrm{K}_0} = 0$, $E_{\mathrm{P}_0} = 1$, and $F_0 = 0$. Note that for these parameters, $\tau_\mathrm{d} >> \tau_\mathrm{o}$. Thus, the pendulum is underdamped and oscillates considerably before damping, similar to SST. The data for training and testing the models were generated numerically using these settings of the pendulum. Training and testing of the models were performed at different instants in time.

We show the results for these best LSTM and NODE models in figure 7. Both models perform accurately and are numerically stable. We observe that the NODE model slightly outperforms the LSTM model. It is interesting to note that the total energy of the pendulum E_T, figure 7(bottom-right), is accurately replicated by both the LSTM and NODE models. E_T was not explicitly modeled by the two time-series ML models, nor were any constraints imposed on the model training to follow the decay of E_T. The values for E_T are computed post-simulation, by summing the kinetic and potential energies from the ML models.

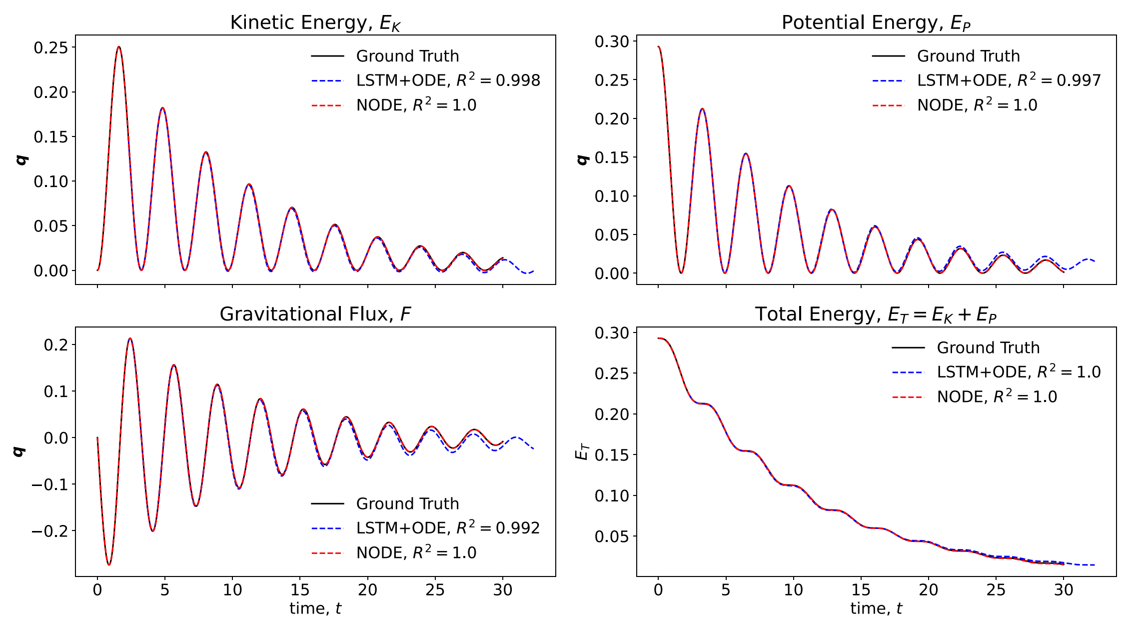

Figure 7. Comparison of the true (analytical) time evolution of the state variables of the damped pendulum with the predictions of the LSTM and NODE models. The error between the true values and ML models is computed using the coefficient of determination, denoted by R^2.

4.1.2. Interpreting the ML model data requirements: minimum information required by model

The interpretability of the data requirements of the time-series ML models is studied by attempting to extract physically realizable timescales from the ML models. For this, the *data hyperparameters*, sequence length and sampling frequency of the input data, are varied to see the effects on the model performance. The objective is to identify the minimum values of the hyperparameters which results in good model performance, thus finding the minimum information that is required by the ML models to effectively learn the system dynamics.

The results of the parametric study of the two ML timescales are shown in figure 8. The data hyperparameters are varied and the corresponding performance of the LSTM and NODE models are observed using 3D contour plots (figures 8(a) and (b)) and their corresponding 2D projections (figure 8(c)). The accuracy of the models (vertical axis in figures 8(a) and (b) and the contour levels in figure 8(c)) is measured by the maximum normalized root mean square error of all the states, $\max_{\boldsymbol{q}}$(NRMS).

For the LSTM model, small values of sampling frequency and sequence length have extremely high errors as noted by the lower-left corner in figures 8(a) and (c). This intuitively makes sense because when enough information is not provided to the model, the model is not able to learn the dynamics of the system. The 2D projection of $\max_{\boldsymbol{q}}$(NRMS) shown in figure 8(c) reveals that this region lives below the minimum information line computed analytically (black dashed curve). The minimum information line corresponds to when the ratio of the sequence length and the sampling frequency is equal to the ratio of the damping timescale, τ_{d}, and oscillating timescale, τ_{o}, of the pendulum. Although there are local regions of high relative error, the general trend is that the LSTM model converges to accurate models beyond the minimum information required to represent the dynamics of the system. Thus, the inherent timescales of the minimum data input for accurate LSTM models corresponds to the physical timescale of the system.

The model error for the NODE models has a spike below the minimum information line similar to LSTM (lower-left corner of figures 8(b) and (c)), whereas closely after the minimum information line, the NODE model performs accurately. Again, these results demonstrate that analyzing the inherent timescales of the input data to the time-series ML model can enable one to extract the timescales of the physical system. This ML-learned timescale correspond to the minimum information required by the time-series ML models to accurately predict the system dynamics.

The NODE model performs notably better than LSTM in the parametric study, as seen from figures 8(a)–(c). Firstly, the highest error observed for the NODE model is an order of magnitude smaller than that of the LSTM model. Furthermore, NODE is much more numerically stable above the minimum information line since the surface plot is much flatter (and smoother) than that of LSTM, which has many local minima and maxima.

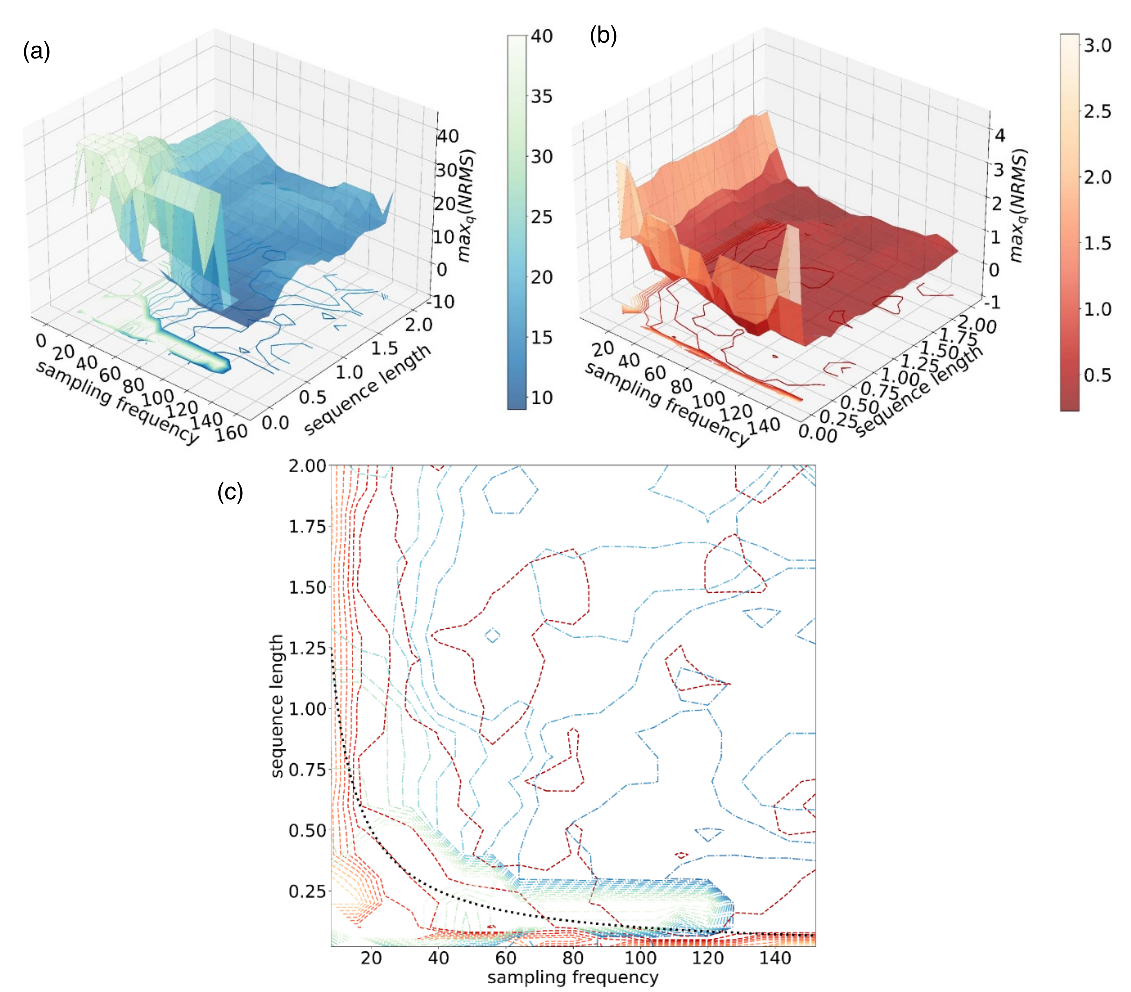

Figure 8. Effect of sequence length and sampling frequency of input data on (a) LSTM model and (b) NODE model. (c) Comparing the time-scales identified by LSTM (blue/green —. contour) and NODE models (red/orange — — contour) based on effect of sequence length and sampling frequency with respect to the analytical timescales (black ··· line denoting minimum information).

4.2. SST initialized with Taylor–Green vortices

We next demonstrate the capability of the time-series ML models to capture the nonlinear dynamics of the energy equations of the SST flow initialized with Taylor–Green vortices. We perform both *a priori* (offline test) and *a posteriori* (online test) testing of the LSTM and NODE models.

4.2.1. A priori testing

The *a priori* or offline test involves testing the accuracy of the models for a given set of inputs. For the LSTM model, we provide values of $\boldsymbol{q}$ at random time instants as inputs to the model and compare the outputs of the model with true or expected values—the time derivative of $\boldsymbol{q}$ (i.e. $\boldsymbol{f}$) at the respective instants. The testing is performed at conditions within and outside the training data regime. The results of the *a priori* test on the LSTM model are shown in figure 9, sorted in the order of the time evolution. We only show the results from the best model identified from the timescales analysis, which will be discussed more in section 4.2.3. The overall accuracy of the model for each variable is measured using the coefficient of determination, R^2, with respect to the true time derivatives of $\boldsymbol{q}$. The results demonstrates the high accuracy of the LSTM model to capture the complex nonlinear relationship between $\boldsymbol{q}$ and its time derivatives. For the NODE model, the outputs are the time evolution of $\boldsymbol{q}$ and thus corresponds to *a posteriori* or online testing.

4.2.2. A posteriori testing

The *a posteriori* or online test involves testing the accuracy and stability of the models over the time evolution of the system. For the LSTM models, this involves coupling the output of the model with the simulations, which in this case is an ODE solver run using a fourth order explicit Runge–Kutta scheme. The outputs from the LSTM model, time derivatives of $\boldsymbol{q}$ at an instant (or the RHS of equation (10), $\boldsymbol{f}$), serves as the RHS function for the ODE solver. We validate the numerical accuracy and stability of the ODE solver by

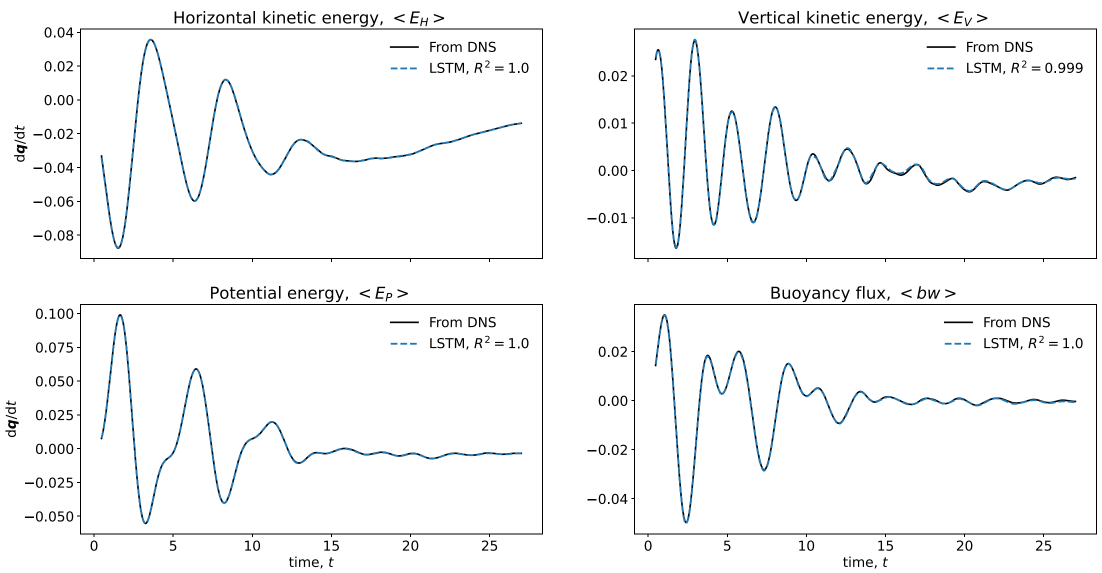

Figure 9. Results from *a priori* tests using an LSTM model. The coefficient of determination, R^2, of the results compared to true outputs are also provided for each variable.

comparing the time evolution of $\boldsymbol{q}$ obtained from DNS with that obtained from the ODE solver when using the true values of time derivative of $\boldsymbol{q}$ as the RHS of the solver.

We show the results from the best LSTM and NODE models (based on the timescales analysis) in figure 10. Both the LSTM and NODE models are able to accurately predict the time evolution of $\boldsymbol{q}$. In terms of the overall accuracy measured by the R^2 values, the NODE model offers better accuracy. Moreover, the predictions are numerically stable over time. Note that although the results show the model initialized from $t = 0$, which is within the training regime, we also tested the model at various initial conditions within and outside the training regime. All the tests resulted in similarly accurate and numerically stable predictions. We wanted to demonstrate that the error acquired by the models when tested in the training regime (see the predictions for E_{P} in the interval $t = [0, 12]$) does not influence the predictions in the testing regime drastically. The time-series models are stable even with accrual of small errors over time, which has been shown to be detrimental for the numerical stability of models focused only on spatial information [65].

The results demonstrate the capability of the time-series ML models to accurately predict the time evolution of complex nonlinear dynamical systems like the reduced-order SST. Similar to the pendulum results, the capability of the ML time-series models to accurately predict the total energy of the turbulent flow is notable (figure 10 bottom row), especially given that no information regarding the total energy was provided to the models. Note that, as a separate training campaign, we also trained the models with a physics-informed approach by imposing constraints of time rate of change of total energy, $\mathrm{d}E_{\mathrm{T}}/\mathrm{d}t$, on the loss functions, thereby explicitly preventing the system from gaining energy. However, these results were not notably different from the current analysis. It is interesting to see the time-series models are able to predict the total energy of the system when the correct timescales are used for the inputs to the ML models.

4.2.3. Interpreting the ML model data hyperparameter for SST

We apply the physical timescale extraction framework discussed in section 4.1.2 on the highly nonlinear SST problem. The current study is motivated by this turbulence modeling problem because there are no analytical solutions for the timescales in turbulent flow. In SST, there are, at a minimum, three important timescales: the buoyancy time scale, the inertial time scale, and the viscous time scale. Currently, the timescales can only be deduced from a combination of analytical work, laboratory or field measurements, and DNS—extremely computationally expensive campaigns, e.g. [5, 66, 67]. Even with these tools, the dominant timescales are hypothesized and then tested against data. This process becomes increasingly difficult when the flow becomes more complex due to, for example, mean shear. Of interest here is whether the relevant timescales could emerge from the ML modeling framework without any hypothesis. The pendulum results demonstrated this claim for a simple dynamical system.

We conduct a similar ML timescales parametric study on the SST data using both the LSTM and NODE models, as shown in figure 11. The parametric plot of timescales identifies unique regions of minimum error: between (1) sampling frequency 30–50 and sequence length 0.4–0.6 for LSTM, and (2) sampling

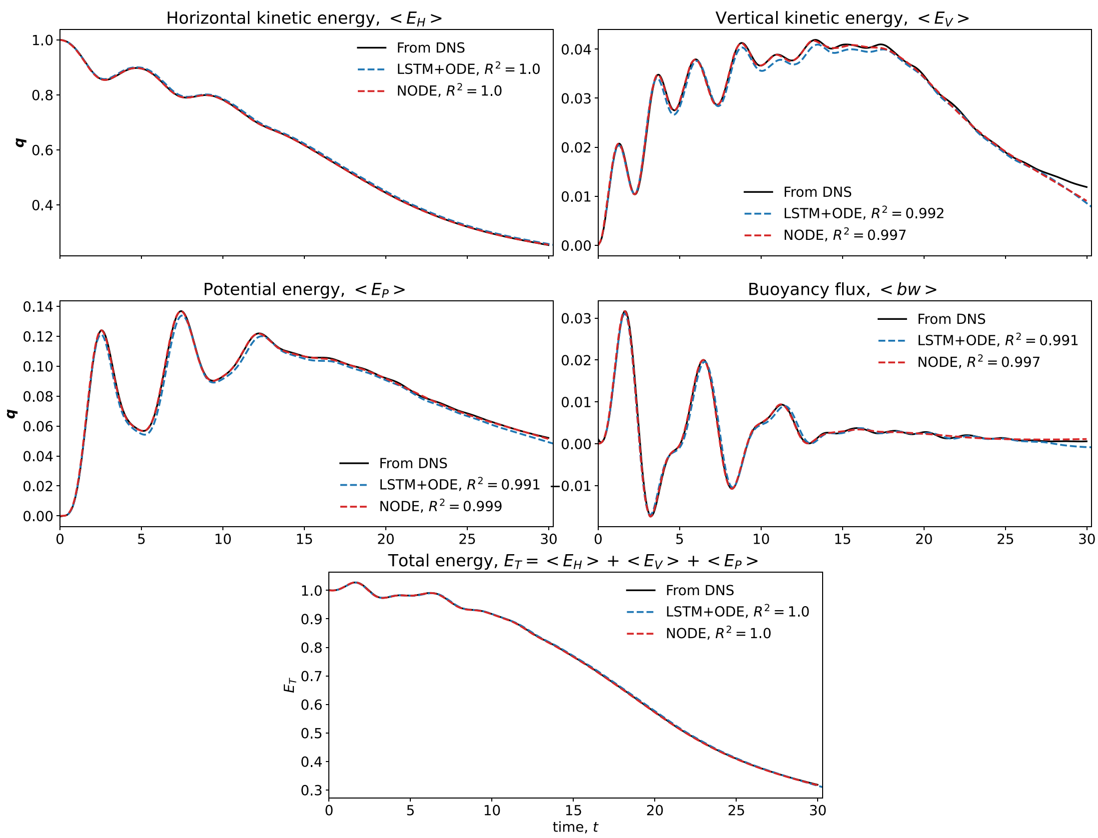

Figure 10. Results from *a posteriori* tests using LSTM and NODE models. The overall accuracy of the models is measured using the coefficient of determination, R^2, of the ML results compared to true time evolution of each variable.

frequency 20–40 and sequence length 0.5–1 for NODE. Interestingly, while the sampling frequency region is similar for both the models, the NODE model is accurate and numerically stable above a sequence length of 0.5. This may pertain to the increased numerical stability and accuracy observed for NODE. These local regions correspond to the minimum information required by the inputs of the ML models to effectively predict the SST dynamics. Based on our inferred knowledge gained from studying the pendulum problem, we can potentially claim that these timescales are related to the physical timescales of the system.

To validate the above claim, we identify the points of lowest error in these regions, denoted by the blue and red dots in figures 11(a) and (b), respectively for the LSTM and NODE models. The ratio of the ML timescales, sequence length to sampling period (1/sampling frequency), at these points of lowest error are approximately 16 for LSTM and 32 for NODE. We note that these ML timescale ratios are integer multiples of the Reynolds number for the SST data, Re = 3200. Physically, the Reynolds number of the SST flow can be defined by the ratio of the fluctuating timescale due to the buoyancy flux and decay timescale due to viscous dissipation [60]. Broadly, they represent the smaller and larger timescales of the system, which are what the time-series ML models extract as well. Thus, the ratio of the ML timescales are integer multiples of the physical timescale ratio known for SST flows. These observations validate our claim that our ML timescales analysis is able to interpret the behavior of the ML models and extract physical timescales of even complex turbulent fluid flow systems.

We generalize the above timescales extraction framework for various parameter ranges of SST flows initialized by Taylor–Green vortices [62]. These flows are parameterized by the Fr and Re. We chose to perform the timescale analysis on two flows: (1) Fr = 2, Re = 3200 and (2) Fr = 4, Re = 6400, thereby testing the timescale extraction framework for change in Fr and Re. The time series of the flow states for the three SST flows are shown in figures 12(a)–(e). Our objective is to compare the timescales extracted using the original/baseline analysis with the dataset Fr = 4 and Re = 3200 (blue lines in figures 12(a)–(e)) with that when (1) the Re is increased (orange lines in figures 12(a)–(e)) and (2) the Fr is decreased (green lines in figures 12(a)–(e)).

We do not change the model hyperparameters of the LSTM and NODE models, and obtain two different models by training the two datasets independently. The focus of generalizability here is not to have a single model to solve parameter ranges of the flow, but to investigate the generalizability of our timescale extraction

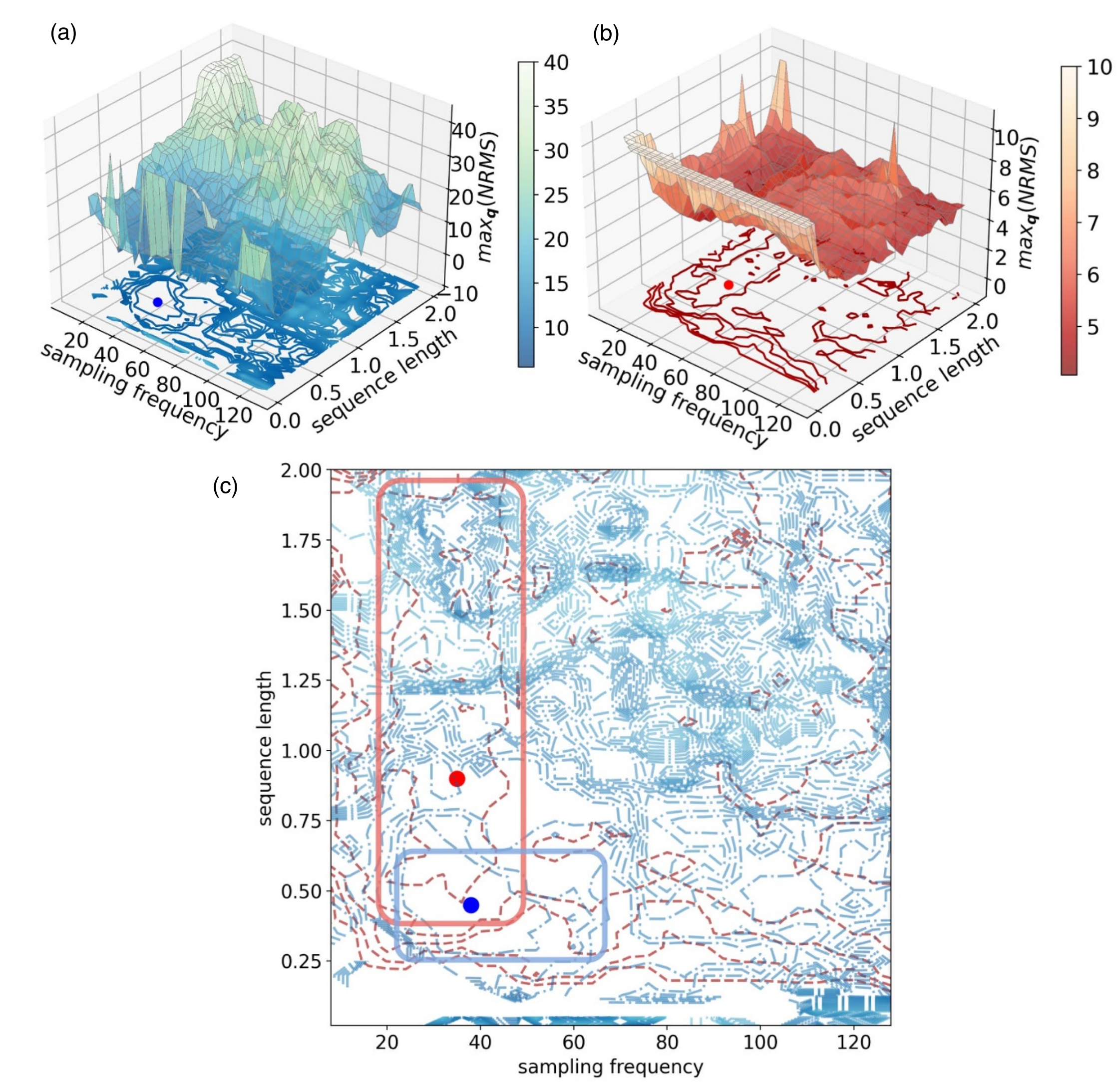

Figure 11. Effect of sequence length and sampling frequency of input data on (a) LSTM model and (b) NODE model. (c) Comparing the time-scales identified by LSTM (blue —. contour) and NODE (red —— contours) models based on effect of sequence length and sampling frequency.

framework. The results of the timescale analyses are shown in figures 12(f)–(h). When the Re is doubled and Fr is kept constant, there is marginal change in the decay rate and buoyancy flux as seen from the time-series in figures 12(a)–(e). The timescales extracted by the ML models also reflect this, depicted by the slight changes in the sampling frequency and sequence length as seen from figure 12(h). When the Fr is halved and Re is unchanged, the flow is characterized by longer times to decay and thus, lower decay rate, as shown in figures 12(a)–(c) and (e). Thus, one could expect the time-series ML models to require more time lengths of data to learn the dynamics—i.e. longer sequence length. This is what is reflected in the timescales extracted by the ML models as shown in figure 12(f). While the sampling frequency remains same, the sequence length increases, denoting the capability of the framework to capture the change in the decay dynamics of the system, thus, demonstrating the generalizability of the timescales extraction framework for various parameter regimes of SST flows.

5. Concluding remarks

We present a time-series based ML modeling framework for second moment closure modeling of URANS simulations of SST. Instead of individually formulating higher-order closures for the nonlinear terms in the URANS equations, we train time-series ML models using high fidelity DNS data to model the entire RHS of the URANS equations. The current approach is applied to one-dimensional SST data, which is spatially reduced by taking advantage of the homogeneity of the flow in three-dimensions. We choose two ML architectures for modeling: (1) LSTM and (2) NODE. The interpretability of the data requirements for the

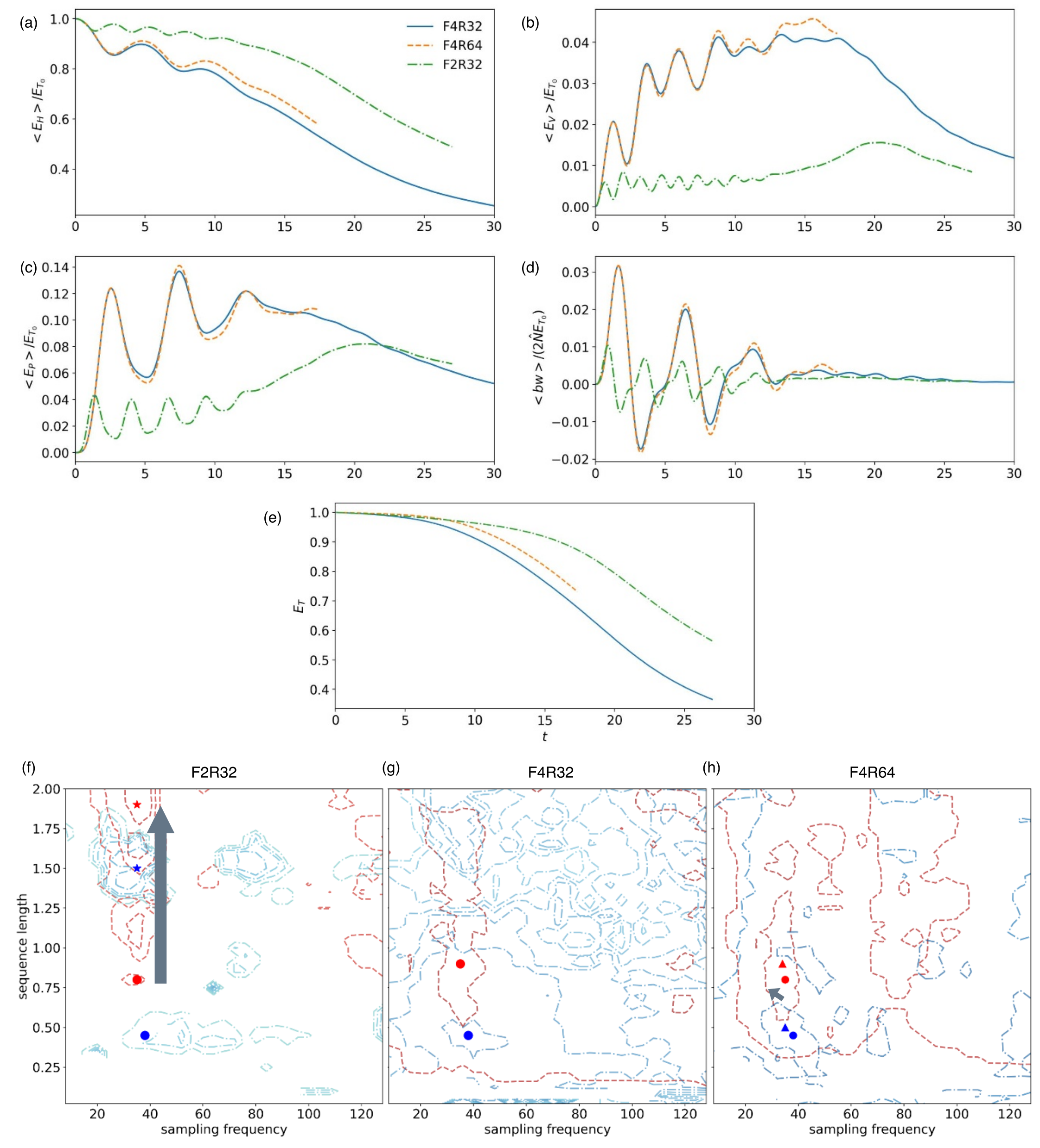

Figure 12. (a)–(e) The time series of SST data at different Fr and Re number values. The abbreviation $F2R32$ denotes the flow initialized with Fr = 2 and Re = 3200, and accordingly for $F4R32$ and $F4R64$. (f)–(h) The parametric study for the LSTM (blue −. contour) and NODE (red −− contour) models conducted independently on the 3 different dataset. The minimum information points are indicated for each case: ★ for Fr = 2, Re = 3200 ($F2R32$); ◯ for Fr = 4, Re = 3200 ($F4R32$; also indicated in the other two cases for comparison); and △ for Fr = 4, Re = 6400 ($F4R64$).

models are assessed by testing the accuracy of the models against two data hyperparameters, the sampling frequency and sequence length of the input data to the models, revealing two ML timescales of the data. The accuracy of the models based on these hyperparameters corresponds to the minimal information required by the ML models to effectively capture the dynamics of the complex system. We find that variations/trends in the ML timescales of the best model corresponds to variations/trends in physical timescales known for such flows.

The framework is first demonstrated for a simple dynamical system—a damped pendulum. The time-series ML models accurately capture the dynamics of the pendulum. Moreover, the ratio of the ML timescales extracted based on the sampling frequency and sequence length of the best models correspond to the ratio of the oscillating frequency to the damping frequency of the pendulum. Physically, this finding denotes that the minimum information required by the time-series ML models to accurately capture the dynamics of a system also captures fundamental timescales of the system.

We extend the demonstration to a decaying SST flow initialized using Taylor–Green vortices. The time series data of the SST involves laminar, transition, and turbulent decay, posing a much more complex nonlinear problem for the time-series ML modeling framework. The LSTM and NODE models are not only accurate for predicting the time evolution of the SST, but also are numerically stable. Moreover, the ratio of the timescales extracted by the ML models is related to the Reynolds number of the flow. We also generalize the timescale extraction framework for various SST, with different Froude and Reynolds numbers. The ML timescales extracted are validated with the relative change in dynamics of the SST according to Froude and Reynolds numbers.

There are a couple of avenues to extend the current framework. First, while the LSTM model has shown great capability to model the dynamics of the SST, there are better time-series models in literature, e.g. GRU [38, 39], which have been shown to be more accurate than LSTM. The current work is a preliminary effort to demonstrate that such time-series models can indeed be used to formulate interpretable models for SST dynamics. There is a clear avenue to extend the analysis for more complex models. Second, the current analysis is not generalized for predicting various parameter ranges of flow using a single model. This requires further investigation and potentially, a framework to incorporate the variation in the nature of the dynamics. Additionally, considerations regarding the amount of data used to train the models and the temporal regimes considered are necessary, as these factors also influence the model's ability to generalize across different parametric regimes of the flow. Finally, while the prediction of the models are deterministic, the behavior of SST in nature warrants uncertainty quantification of the results of the models. The unique opportunity provided by GPU enabled high-performance computing platforms to perform ML inference at scale can further enable probabilistic approaches for modeling. Such complimentary advancements to the current framework can enable modeling of flows with higher complexity and spatial dimensions.

Data availability statement

The specific codes and data that support the findings of this study are openly available in Zenodo [68] at https://doi.org/10.5281/zenodo.13787850.

Acknowledgments

This research was supported by the Office of Naval Research via Grant N00014-19-1-2152. High performance computing resources were provided via the U.S. Department of Energy (DOE) INCITE program by the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. Computing resources were also provided through the U.S. Department of Defense High Performance Computing Modernization Program.

Author contributions

M G M, S M deB K, and J J R initiated and designed the project. S M deB K curated the stably stratified turbulence direct numerical simulation data. M G M, D L, and A D S designed and performed the machine learning analysis with feedback from A K and W H B. M G M, D L, A D S, and S M deB K wrote the manuscript with feedback from all the coauthors.

Conflict of interest

The authors declare no competing interests.

ORCID iDs

Muralikrishnan Gopalakrishnan Meena https://orcid.org/0000-0003-4048-4639
Demetri Liousas https://orcid.org/0009-0002-3483-8867
Andrew D Simin https://orcid.org/0000-0003-4569-9049
Aditya Kashi https://orcid.org/0000-0003-2589-3792
Wesley H Brewer https://orcid.org/0000-0002-3639-3956
James J Riley https://orcid.org/0000-0002-9497-0766
Stephen M de Bruyn Kops https://orcid.org/0000-0002-7727-8786

References

[1] Kimura Y and Herring J R 1996 Diffusion in stably stratified turbulence *J. Fluid Mech.* **328** 253–69
[2] Lindborg E 2006 The energy cascade in a strongly stratified fluid *J. Fluid Mech.* **550** 207–42
[3] Kumar A, Chatterjee A G and Verma M K 2014 Energy spectrum of buoyancy-driven turbulence *Phys. Rev.* E **90** 023016

[4] Riley J J and Lindborg E 2012 Recent progress in stratified turbulence *Ten Chapters in Turbulence* ed P A Davidson, Y Kaneda and K R Sreenivasan (Cambridge University Press) pp 269–317

[5] de Bruyn Kops S M and Riley J J 2019 The effects of stable stratification on the decay of initially isotropic homogeneous turbulence *J. Fluid Mech.* **860** 787–821

[6] Mater B D and Venayagamoorthy S K 2014 A unifying framework for parameterizing stably stratified shear-flow turbulence *Phys. Fluids* **26** 036601

[7] Norman M R *et al* 2021 Unprecedented cloud resolution in a GPU-enabled full-physics atmospheric climate simulation on OLCF's summit supercomputer *Int. J. High Perfom. Comput. Appl.* **36** 93–105

[8] Jain N, Pham H T, Huang X, Sarkar S, Yang X and Kunz R 2022 Second moment closure modeling and direct numerical simulation of stratified shear layers *J. Fluids Eng.* **144** 041102

[9] Jain N, Huang X, Yang X and Kunz R 2022 A study of second moment closure modeling for stratified wakes using DNS ensembles *Fluids Engineering Division Summer Meeting* vol 85833 (American Society of Mechanical Engineers) p V001T03A012

[10] Brunton S L, Noack B R and Koumoutsakos P 2020 Machine learning for fluid mechanics *Annu. Rev. Fluid Mech.* **52** 477–508

[11] Duraisamy K, Iaccarino G and Xiao H 2019 Turbulence modeling in the age of data *Annu. Rev. Fluid Mech.* **51** 357–77

[12] Rasp S, Pritchard M S and Gentine P 2018 Deep learning to represent subgrid processes in climate models *Proc. Natl Acad. Sci.* **115** 9684–9689

[13] Yuval J and Gorman P A O' 2020 Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions *Nat. Commun.* **11** 1–10

[14] Krasnopolsky V M, Fox-Rabinovitz M S and Belochitski A A 2013 Using ensemble of neural networks to learn stochastic convection parameterizations for climate and numerical weather prediction models from data simulated by a cloud resolving model *Adv. Artif. Neural Syst.* **2013** 485913

[15] Gentine P, Pritchard M, Rasp S, Reinaudi G and Yacalis G 2018 Could machine learning break the convection parameterization deadlock? *Geophys. Res. Lett.* **45** 5742–51

[16] Brenowitz N D and Bretherton C S 2019 Spatially extended tests of a neural network parametrization trained by coarse-graining *J. Adv. Model. Earth Syst.* **11** 2728–44

[17] Watt Meyer O, Brenowitz N D, Clark S K, Henn B, Kwa A, McGibbon J, Perkins W A and Bretherton C S 2021 Correcting weather and climate models by machine learning nudged historical simulations *Geophys. Res. Lett.* **48** e2021GL092555

[18] Bretherton C S, Henn B, Kwa A, Brenowitz N D, Watt-Meyer O, McGibbon J, Perkins W A, Clark S K and Harris L 2022 Correcting coarse-grid weather and climate models by machine learning from global storm-resolving simulations *J. Adv. Model. Earth Syst.* **14** e2021MS002794

[19] Cheng Y, Giometto M G, Kauffmann P, Lin L, Cao C, Zupnick C, Li H, Li Q, Huang Y and Abernathey R *et al* 2022 Deep learning for subgrid-scale turbulence modeling in large-eddy simulations of the convective atmospheric boundary layer *J. Adv. Model. Earth Syst.* **14** e2021MS002847

[20] Salehipour H and Richard Peltier W 2019 Deep learning of mixing by two 'atoms' of stratified turbulence *J. Fluid Mech.* **861** R4

[21] Huang X L D, Kunz R F and Yang X I A 2023 Linear logistic regression with weight thresholding for flow regime classification of a stratified wake *Theor. Appl. Mech. Lett.* **13** 100414

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

[23] Oda T, Okayasu S, Hara N, Tanimoto K, Haghiri A, Xu X and Sandberg R 2022 Study on the applicability of a machine-learning framework to improve modeling for the stratification phenomenon *Int. Conf. on Nuclear Engineering* vol 86434 (American Society of Mechanical Engineers) p V008T08A026

[24] Xu X, Haghiri A, Sandberg R D, Oda T and Tanimoto K 2024 Data-driven turbulence modelling of inherently unsteady flow in stratified water storage tanks *Int. J. Heat Mass Transfer* **219** 124854

[25] Bolbasova L A, Andrakhanov A A and Yu Shikhovtsev A 2021 The application of machine learning to predictions of optical turbulence in the surface layer at Baikal astrophysical observatory *Mon. Not. R. Astron. Soc.* **504** 6008–17

[26] Couchman M M P, Wynne-Cattanach B, Alford M H, Caulfield C-cille P, Kerswell R R, MacKinnon J A and Voet G 2021 Data-driven identification of turbulent oceanic mixing from observational microstructure data *Geophys. Res. Lett.* **48** e2021GL094978

[27] Lewin S F, de Bruyn Kops S M, Caulfield Colm-cille P and Portwood G D 2023 A data-driven method for modelling dissipation rates in stratified turbulence *J. Fluid Mech.* **977** A37

[28] Lefauve A and Couchman M M P 2024 Data-driven classification of sheared stratified turbulence from experimental shadowgraphs *Phys. Rev. Fluids* **9** 034603

[29] Petropoulos N, Couchman M M P, Mashayek A, de Bruyn Kops S M and Caulfield C-cille P 2024 Prandtl number effects on extreme mixing events in forced stratified turbulence *J. Fluid Mech.* **983** R1

[30] Lefauve A and Couchman M M P Routes to stratified turbulence revealed by unsupervised classification of experimental data 2023 (arXiv:2305.04051)

[31] Connor J T, Douglas Martin R and Atlas L E 1994 Recurrent neural networks and robust time series prediction *IEEE Trans. Neural Netw.* **5** 240–54

[32] Yu Y, Si X, Hu C and Zhang J 2019 A review of recurrent neural networks: LSTM cells and network architectures *Neural Comput.* **31** 1235–70

[33] Che Z, Purushotham S, Cho K, Sontag D and Liu Y 2018 Recurrent neural networks for multivariate time series with missing values *Sci. Rep.* **8** 6085

[34] Hewamalage H, Bergmeir C and Bandara K 2021 Recurrent neural networks for time series forecasting: current status and future directions *Int. J. Forecast.* **37** 388–427

[35] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80

[36] Hua Y, Zhao Z, Li R, Chen X, Liu Z and Zhang H 2019 Deep learning with long short-term memory for time series prediction *IEEE Commun. Mag.* **57** 114–9

[37] Lindemann B, Müller T, Vietz H, Jazdi N and Weyrich M 2021 A survey on long short-term memory networks for time series prediction *Proc. CIRP* **99** 650–5

[38] Chung J, Gulcehre C, Cho K and Bengio Y 2014 Empirical evaluation of gated recurrent neural networks on sequence modeling (arXiv:1412.3555)

[39] Weerakody P B, Wai Wong K, Wang G and Ela W 2021 A review of irregular time series data handling with gated recurrent neural networks *Neurocomputing* **441** 161–78

[40] Bukka S R, Gupta R, Magee A R and Jaiman R K 2021 Assessment of unsteady flow predictions using hybrid deep learning based reduced-order models *Phys. Fluids* **33** 013601

[41] Drikakis D, William Kokkinakis I, Fung D and Michael Spottswood S 2024 Generalizability of transformer-based deep learning for multidimensional turbulent flow data *Phys. Fluids* **36** 026102

[42] Arras L, Arjona-Medina J, Widrich M, Montavon G, Gillhofer M, Müller K-R, Hochreiter S and Samek W 2019 Explaining and interpreting LSTMs *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* pp 211–38

[43] Guo T, Lin T and Antulov-Fantulin N 2019 Exploring interpretable LSTM neural networks over multi-variable data *Int. Conference on Machine Learning* (PMLR) pp 2494–504 (available at: https://proceedings.mlr.press/v97/guo19b.html)

[44] Ding Y, Zhu Y, Feng J, Zhang P and Cheng Z 2020 Interpretable spatio-temporal attention LSTM model for flood forecasting *Neurocomputing* **403** 348–59

[45] Lim B, Arık S O, Loeff N and Pfister T 2021 Temporal fusion transformers for interpretable multi-horizon time series forecasting *Int. J. Forecast.* **37** 1748–64

[46] Riley J J and Lelong M-P 2000 Fluid motions in the presence of strong stable stratification *Annu. Rev. Fluid Mech.* **32** 613–57

[47] Kumar A, Verma M K and Sukhatme J 2017 Phenomenology of two-dimensional stably stratified turbulence under large-scale forcing *J. Turbul.* **18** 219–39

[48] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44

[49] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66

[50] Chen R T Q, Rubanova Y, Bettencourt J and Duvenaud D K 2018 9Neural ordinary differential equations *Advances in Neural Information Processing Systems* p 31 (available at: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf)

[51] Legaard C, Schranz T, Schweiger G, Drgoňa J, Falay B, Gomes C, Iosifidis A, Abkar M and Larsen P 2023 Constructing neural network based models for simulating dynamical systems *ACM Comput. Surv.* **55** 1–34

[52] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* pp 770–8 (available at: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf)

[53] Dhadphale J M, Unni V R, Saha A and Sujith R I 2022 Neural ODE to model and prognose thermoacoustic instability *Chaos* **32** 013131

[54] Kang S and Constantinescu E M 2023 Learning subgrid-scale models with neural ordinary differential equations *Comput. Fluids* **261** 105919

[55] Linot A J, Burby J W, Tang Q, Balaprakash P, Graham M D and Maulik R 2023 Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems *J. Comput. Phys.* **474** 111838

[56] Linot A J and Graham M D 2022 Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations *Chaos* **32** 073110

[57] Zeng K, Linot A J and Graham M D 2022 Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning *Proc. R. Soc.* A **478** 20220297

[58] Linot A J, Zeng K and Graham M D 2023 Turbulence control in plane Couette flow using low-dimensional neural ODE-based models and deep reinforcement learning *Int. J. Heat Fluid Flow* **101** 109139

[59] Rahman A, Drgoňa J, Tuor A and Strube J 2022 Neural ordinary differential equations for nonlinear system identification 2022 *American Control Conf. (ACC)* pp 3979–84

[60] Riley J J and de Bruyn Kops S M 2003 Dynamics of turbulence strongly influenced by buoyancy *Phys. Fluids* **15** 2047–59

[61] Hebert D A and de Bruyn Kops S M 2006 Relationship between vertical shear rate and kinetic energy dissipation rate in stably stratified flows *Geophys. Res. Lett.* **33** L06602

[62] Hebert D A and de Bruyn Kops S M 2006 Predicting turbulence in flows with strong stable stratification *Phys. Fluids* **18** 1–10

[63] Nelson R A and Olsson M G 1986 The pendulum—rich physics from a simple system *Am. J. Phys.* **54** 112–21

[64] Liousas D, Simin A D, Kashi A, Brewer W H, de Bruyn Kops S M and Gopalakrishnan Meena M 2024 Predicting timescales in dynamical systems with explainable machine learning models *In XAI4Sci (Explainable Machine Learning for Sciences) Workshop Proc. 38th Annual AAAI Conf. on Artificial Intelligence* (AAAI) p 10 (available at: https://xai4sci.github.io/papers/2024/10_predicting_timescales_in_dynamical_systems_with_explainable_machine_learning_models.pdf)

[65] Gopalakrishnan Meena M, Norman M R, Hall D M and Pritchard M S 2024 Spatially local surrogate modeling of subgrid-scale effects in idealized atmospheric flows: a deep learned approach using high-resolution simulation data *Artif. Intell. Earth Syst.* **3** e230043

[66] Riley J J, Couchman M M P and de Bruyn Kops S M 2023 The effect of Prandtl number on decaying stratified turbulence *J. Turbul.* **24** 330–48

[67] Couchman M M P, de Bruyn Kops S M and Caulfield Colm-cille P 2023 Mixing across stable density interfaces in forced stratified turbulence *J. Fluid Mech.* **961** A20

[68] Gopalakrishnan Meena M, Liousas D, Simin A and de Bruyn Kops S M 2024 sst-urans-timeseries-ml: Machine-learned closure of URANS for stably stratified turbulence *Zenodo* (https://doi.org/10.5281/zenodo.13787850)