Journal of Machine Learning for Modeling and Computing, 5(2):73–111 (2024)

PREDICTING FLUID PARTICLE TRAJECTORIES WITHOUT FLOW COMPUTATIONS: A DATA-DRIVEN APPROACH

Jianchen Wei,[1] Melissa A. Green,[2] Lixin Shen,[1] & Minghao W. Rostami[3,*]

[1]*Department of Mathematics, Syracuse University, 215 Carnegie Building, Syracuse, New York 13244, USA*

[2]*Aerospace Engineering & Mechanics, University of Minnesota, Minneapolis, Minnesota 55455, USA*

[3]*Department of Mathematics and Statistics, Binghamton University, PO Box 6000, Binghamton, New York 13902-6000, USA*

*Address all correspondence to: Minghao W. Rostami, Department of Mathematics and Statistics, Binghamton University, PO Box 6000, Binghamton, New York 13902-6000, USA, E-mail: mrostami@binghamton.edu

Original Manuscript Submitted: 12/11/2023; Final Draft Received: 4/12/2024

The Lagrangian analysis of a fluid flow entails calculating the trajectories of fluid particles, which are governed by an autonomous or non-autonomous dynamical system, depending on whether the flow is steady or unsteady. In conventional methods, a particle's position is incremented time step by time step using a numerical solver for ordinary differential equations (ODEs), assuming that the fluid velocity field is known analytically or can be acquired through either numerical simulation or experimentation. In this work, we assume instead that the velocity field is unavailable but abundant trajectory data are available. Leveraging the data processing power of deep neural networks, we construct data-driven models for the increment in particles' positions and simulate their trajectories by applying such a model recursively. We develop a novel, more experiment-friendly model for non-autonomous systems and compare it with two existing models: one developed for autonomous systems only and one developed for non-autonomous systems with some knowledge of the time-varying terms. Theoretical analysis is performed for all three that sheds a new light on the existing models. Numerical results obtained for several benchmark problems confirm the validity of these models for advancing fluid particles' positions and reveal how their performance depends on the structure of the neural network and physical features of the flow, such as vortices.

KEY WORDS: *fluid particle trajectory, dynamical system, steady flow, unsteady flow, data-driven model, deep neural network*

2689–3967/24/$35.00

1. INTRODUCTION

In the analysis of fluid mechanic systems, there are two main frameworks: the Eulerian and the Lagrangian. In the Eulerian, a flow field is specified by the evolution of fluid properties at specific positions and times. In the Lagrangian, the dynamics are specified along trajectories of defined fluid tracer points. At any time point, the position of each Lagrangian tracer is governed by a system of ordinary differential equations (ODEs) that equates the instantaneous rate of change in the particle's position with the local fluid velocity. For a steady flow, the fluid velocity field does not vary with time, resulting in an autonomous dynamical system, whereas an unsteady flow, characterized by a time-varying fluid velocity field, gives rise to a non-autonomous system.

Conventional methods for simulating fluid particle trajectories apply a numerical ODE solver, such as the forward Euler method or a Runge-Kutta method, to the system of ODEs to advance a particle's position time step by time step, requiring the resolution of the local fluid velocity at every step. In applications where the fluid velocity field is not known analytically, resolving it is the most computationally intensive component of fluid particle tracking. One approach is to first use a system of partial differential equations (PDEs), such as the Navier-Stokes equations, to model the fluid dynamics and then calculate the fluid velocity by solving this system of PDEs numerically (Elman et al., 2014). It assumes that a sufficiently accurate model already exits or can be derived from first principles and that a sufficiently accurate and computationally efficient numerical method already exists or can be developed to simulate this model. Besides numerical simulation, the fluid velocity field can also be obtained through experimentation using flow measurement techniques such as the particle image velocimetry (PIV) (Raffel et al., 2018).

In this work, we consider a completely different paradigm for tracking fluid particles, where the fluid velocity field is unavailable but plenty of particle trajectory data are available instead. This is common in the use of particle tracking velocimetry (PTV), an experimental technique similar to PIV that can generate complete trajectories of individual particles (Schanz et al., 2016). Lagrangian analysis of fluid trajectory dynamics is also of major interest in fluids, dynamical systems, and oceanography (Haller, 2015; van Sebille et al., 2018). That is, we aim to predict fluid particle trajectories based on these data without any knowledge of the fluid velocity field. We use a deep neural network (DNN) (Goodfellow et al., 2016), which is a composition of layers of affine functions and non-linear functions, to model an increment function that produces the change in a particle's position over a small time interval based on its position at the start of the interval. It is trained so that the model parameters best fit the trajectory data. Applying the trained DNN recursively, we can then compose the entire trajectory of a fluid particle over a much larger time domain. Our work is closely related to Qin et al. (2019, 2021), where DNNs were used to model the solutions to systems of ODEs whose forms are only partially known at best. Due to the incomplete equations, direct simulation using a numerical ODE solver is infeasible. The existence of increment functions for autonomous systems and non-autonomous systems was shown in Qin et al. (2019, 2021), respectively; several ResNet (He et al., 2016)-like DNN models were developed to approximate a flow map associated with such a system, which is, loosely speaking, the sum of an increment function and the identity function. It produces the future state of the system at the end of a small time interval given its current state at the start of the interval. The framework developed in Qin et al. (2019, 2021) has already been applied to learn the solutions to biological models (Su et al., 2021), PDEs (Chen et al., 2022; Wu and Xiu, 2020), and reduced systems (Fu et al., 2020).

However, there is a major disadvantage of using the increment function in Qin et al. (2021) for the prediction of non-autonomous systems purely from measured data, such as in the fluid

dynamic or oceanographic examples mentioned above. Learning it requires knowing how the forcing terms that drive the evolution of the system state depend on time, that is, the forcing terms need to be partially known. In the context of tracking fluid particles, this means knowing how the fluid velocity field of an unsteady flow varies with time, which is unrealistic in problems where the fluid velocity is truly unknown. Our contributions are as follows. First, we propose a new increment function for non-autonomous systems that can be learned without any knowledge of the time-dependent terms, allowing for the solution of these systems in applications where only trajectory data are available. Secondly, we find novel analytic expressions of all three increment functions, including the one proposed here and the two existing ones in Qin et al. (2019, 2021), that complement the analysis in Qin et al. (2019, 2021) and shed a new light on these functions. Thirdly, we validate the applicability of DNN models built for the three increment functions at tracking fluid particles in several benchmark problems. We also compare the performance of these models and examine how it can be influenced by network structures and flow features. While the aim here and in Qin et al. (2019, 2021) is to learn the solution to a system of ODEs without knowing its complete form, there is also a large body of work on data-driven equation discovery, such as Raissi et al. (2018), Zhuang et al. (2021), Proctor et al. (2016), Long et al. (2018, 2019), Lu et al. (2021), and Lin et al. (2023), where the goal is to uncover the governing physical laws, in the form of ODEs or PDEs, hidden in data. The network structures developed in Qin et al. (2019, 2021) have also been used to learn Hamiltonian systems (Wu et al., 2020) and perform model corrections (Chen and Xiu, 2021). Our work is also among many that apply machine learning techniques to solve fluid mechanics problems [see Brunton et al. (2020) for an overview].

We note that in this new paradigm, the challenges of predicting fluid particle trajectories arise from acquiring trajectory data and training a DNN model for the increment function or flow map, instead of from resolving the fluid velocity field as in the traditional paradigm. The accuracy of trajectory calculation depends on the accuracy of the DNN model, which in turn depends on the quality and quantity of the data as well as the structure of the NN and the optimality of its parameters, instead of on the accuracy of the time-stepping scheme and the mathematical modeling, numerical simulation, and experimental measurement of the fluid velocity field as in the traditional paradigm.

The rest of the paper is organized as follows. In Section 2, Appendix A, and Appendix B, we derive a new increment function for non-autonomous systems and find novel analytic expressions for the two existing increment functions in Qin et al. (2019, 2021). In Section 3, we build and compare DNN models for the increment functions corresponding to a variety of fluid flows. We also apply them recursively to predict full particle trajectories. The advantages and disadvantages of the proposed and existing increment functions for non-autonomous systems are discussed in both Sections 2 and 3. We summarize and conclude the paper in Section 4.

2. METHODS

In the study of fluid dynamics, unsteady flows are characterized by time-dependent velocity fields, whereas steady flows maintain the same velocity field over time. Accordingly, positions of particles immersed in an unsteady flow must be modeled using non-autonomous dynamical systems, whereas positions of particles immersed in a steady flow can be described by autonomous dynamical systems.

For either type of flows, we aim to construct a neural network (NN) to model the increments in fluid particles' positions over a fixed, relatively small time duration. In the case of a steady

flow, the increment is a function of the starting position only, whereas in the case of an unsteady flow, it depends on the starting time point as well. In Section 2.1, we derive three variants of the increment: one for steady flows, and two for unsteady flows. Our derivation, which is different from the one in Qin et al. (2019, 2021), allows us to find an analytic expression for each variant in terms of the fluid velocity vector field. In Section 2.2, we present the NN for modeling the increment, and an algorithm for predicting particle trajectories using this model. The data sets used to train and test the model are described in Section 2.3.

2.1 The Δ-Increment and Flow Map

2.1.1 Steady Flows

In the case of a steady flow, let $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^{d_{\text{flo}}}$ denote the fluid velocity vector at location $\boldsymbol{x} \in \mathbb{R}^{d_{\text{flo}}}$, where d_{flo} is the dimension of the flow and is 2 or 3 in this work. Let $\mathbf{x}(t) \in \mathbb{R}^{d_{\text{flo}}}$ denote the position of a particle in this flow at time t. Assuming that the particle moves at the local fluid velocity (the no-slip condition), $\mathbf{x}(t)$ satisfies the following system of autonomous ODEs:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \boldsymbol{u}(\mathbf{x}(t)), \quad t \in [0, T], \tag{1}$$

where $T > 0$. For this system, we can prove the following.

Theorem 1. *In Eq.* (1)*, let the expression of* $\boldsymbol{u}(\boldsymbol{x})$ *in terms of* $\boldsymbol{x}$ *be fixed. Let* $0 < \Delta < T$ *be fixed. Then there exists a function* $\boldsymbol{\Phi}_\Delta$ *from* $\mathbb{R}^{d_{flo}}$ *to* $\mathbb{R}^{d_{flo}}$ *such that for any* $\mathbf{x}$ *satisfying Eq.* (1) *and any* $t \in [0, T - \Delta]$*,*

$$\boldsymbol{\Phi}_\Delta(\mathbf{x}(t)) = \mathbf{x}(t + \Delta) - \mathbf{x}(t). \tag{2}$$

The proof of this theorem can be found in Appendix A. For the rest of the paper, we refer to $\boldsymbol{\Phi}_\Delta$ and its counterparts for unsteady flows as Δ-increments. We also refer to $\boldsymbol{\Phi}_\Delta$ as the steady Δ-increment to distinguish it from a Δ-increment for an unsteady flow, which will be represented by $\boldsymbol{\Psi}_\Delta$ or $\boldsymbol{\Xi}_\Delta$ in later sections. This concept was first introduced in Qin et al. (2019), where the existence of $\boldsymbol{\Phi}_\Delta$ was shown using the fundamental theorem of calculus and mean-value theorem. Our proof is based on the Taylor expansion of $\mathbf{x}(t)$ instead and has the advantage of being constructive, that is, we are able to find an expression of $\boldsymbol{\Phi}_\Delta$ in terms of the fluid velocity $\boldsymbol{u}$ (see Appendix B).

Accordingly, the flow map for the autonomous system Eq. (1) can be written as

$$\boldsymbol{\Phi}_\Delta(\mathbf{x}(t)) = \mathbf{x}(t) + \boldsymbol{\Phi}_\Delta(\mathbf{x}(t)), \quad t \in [0, T - \Delta]. \tag{3}$$

We note the difference between the Δ-increment $\boldsymbol{\Phi}_\Delta$ and the flow map $\boldsymbol{\Phi}_\Delta$: given a starting position of a particle, the former gives the *change in the position* of the particle whereas the latter gives *the actual position* of the particle, at the end of a time duration of length Δ.

2.1.2 Unsteady Flows

In the case of an unsteady flow, let $\boldsymbol{u}(\boldsymbol{x}, \gamma(t)) \in \mathbb{R}^{d_{\text{flo}}}$ denote the fluid velocity vector at location $\boldsymbol{x} \in \mathbb{R}^{d_{\text{flo}}}$ and time t, where γ is a function from $[0, T]$ to $\mathbb{R}$ that specifies the dependence of $\boldsymbol{u}$

on time.† Again assuming the no-slip condition, the position of a particle in this flow at time t, $\mathbf{x}(t)$, satisfies the following system of non-autonomous ODEs:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \boldsymbol{u}(\mathbf{x}(t), \gamma(t)),\ t \in [0, T]. \tag{4}$$

For this system, we can prove the following.

Theorem 2. *In Eq.* (4)*, let the expression of* $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ *in terms of* $\boldsymbol{x}$ *and* γ *be fixed. Let* $0 < \Delta < T$ *be fixed. Then there exists a function* $\boldsymbol{\psi}_\Delta$ *from* $\mathbb{R}^{d_{fo}+m+1}$ *to* $\mathbb{R}^{d_{fo}}$ *such that for any* $\mathbf{x}$*, any* m*th-degree polynomial* γ *satisfying Eq.* (4)*, and any* $t \in [0, T-\Delta]$*,*

$$\boldsymbol{\psi}_\Delta(\mathbf{x}(t), \boldsymbol{\Gamma}(t)) = \mathbf{x}(t+\Delta) - \mathbf{x}(t), \tag{5}$$

where $\boldsymbol{\Gamma}(t) = \begin{bmatrix}\gamma(t) & \gamma^{(1)}(t) & \cdots & \gamma^{(m)}(t)\end{bmatrix}^T \in \mathbb{R}^{m+1}$ *and* $\gamma^{(k)}$ *denotes the* k*th derivative of* γ*.*

The proof is very similar to the proof of Theorem 1 after rewriting the non-autonomous system Eq. (4) into the following autonomous system: for $t \in [0, T]$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}\\ \boldsymbol{\Gamma}\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}(\mathbf{x}, \gamma(t))\\ \begin{bmatrix}\gamma^{(1)}(t) & \gamma^{(2)}(t) & \cdots & \gamma^{(m)}(t) & 0\end{bmatrix}^T\end{bmatrix}. \tag{6}$$

Our proof is again different from the existing one in Qin et al. (2021), where the existence of the Δ-increment $\boldsymbol{\psi}_\Delta$ was first shown. It gives an explicit expression of $\boldsymbol{\psi}_\Delta$ in terms of $\boldsymbol{u}$ (see Appendix B). We refer to this variant of Δ-increment as the γ-explicit unsteady Δ-increment to emphasize that the time-dependent term $\gamma(t)$ is known. The flow map of the non-autonomous system Eq. (4) can therefore be written as

$$\boldsymbol{\Psi}_\Delta(\mathbf{x}(t), \boldsymbol{\Gamma}(t)) = \mathbf{x}(t) + \boldsymbol{\psi}_\Delta(\mathbf{x}(t), \boldsymbol{\Gamma}(t)), \qquad t \in [0, T-\Delta]. \tag{7}$$

In the case where γ is not a polynomial, for any $t \in [0, T-\Delta]$, if we can find an mth-degree polynomial ρ that approximates γ globally on $[0, T]$ or locally in the neighborhood $[t, t+\Delta]$, then instead of Eqs. (5) and (7), we have

$$\boldsymbol{\psi}_\Delta(\mathbf{x}(t), \boldsymbol{P}(t)) \approx \mathbf{x}(t+\Delta) - \mathbf{x}(t), \tag{8}$$

$$\boldsymbol{\Psi}_\Delta(\mathbf{x}(t), \boldsymbol{P}(t)) = \mathbf{x}(t) + \boldsymbol{\psi}_\Delta(\mathbf{x}(t), \boldsymbol{P}(t)), \tag{9}$$

where $\boldsymbol{P}(t) = \begin{bmatrix}\rho(t) & \rho^{(1)}(t) & \cdots & \rho^{(m)}(t)\end{bmatrix}^T$ and $\rho^{(k)}$ denotes the kth derivative of ρ.

Alternatively, we propose to rewrite the non-autonomous system Eq. (4) into the the following different autonomous system: for $t \in [0, T]$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}\\ t\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}(\mathbf{x}, \gamma(t))\\ 1\end{bmatrix}. \tag{10}$$

We can then prove the following for Eq. (4) by applying an argument similar to the proof of Theorem 1 to the autonomous system Eq. (10).

†There could be more than one time-dependent terms $\gamma_1, \gamma_2, \cdots$ in $\boldsymbol{u}$. The changes to Theorem 2 and its proof are only technical. Therefore, we only consider the special case (4) in this paper.

Theorem 3. *In Eq.* (4)*, let the expression of* $\gamma(t)$ *in terms of* t *and the expression of* $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ *in terms of* $\boldsymbol{x}$ *and* γ *be fixed. Let* $0 < \Delta < T$ *be fixed. Then there exists a function* $\boldsymbol{\xi}_\Delta$ *from* $\mathbb{R}^{d_{flo}+1}$ *to* $\mathbb{R}^{d_{flo}}$ *such that for any* $\mathbf{x}$ *satisfying Eq.* (4) *and any time point* $t \in [0, T-\Delta]$*,*

$$\boldsymbol{\xi}_\Delta(\mathbf{x}(t), t) = \mathbf{x}(t+\Delta) - \mathbf{x}(t). \tag{11}$$

The expression of $\boldsymbol{\xi}_\Delta$ in terms of $\boldsymbol{u}$ can be found in Appendix B as well. We refer to this variant of Δ-increment as the γ-implicit unsteady Δ-increment to emphasize that the time-dependent term $\gamma(t)$ is unknown. Therefore, besides Eq. (7), the flow map of the nonautonomous system Eq. (4) can also be rewritten as

$$\boldsymbol{\Xi}_\Delta(\mathbf{x}(t), t) = \mathbf{x}(t) + \boldsymbol{\xi}_\Delta(\mathbf{x}(t), t), \quad t \in [0, T-\Delta]. \tag{12}$$

We have seen that for the non-autonomous system Eq. (4), both the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ and the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ can be used to calculate the change in a particle's position. We summarize the advantages and disadvantages of the two below.

— To learn $\boldsymbol{\psi}_\Delta$ from data, we must know γ or, in cases where γ is not a polynomial, its polynomial approximation, ρ. This is unrealistic when an expression of the fluid velocity, $\boldsymbol{u}$, is truly unknown and only trajectory data are available. Learning $\boldsymbol{\xi}_\Delta$ only requires knowing the trajectory data and the associated "time stamps," which makes the data collection considerably simpler and "experiment-friendly."

— Once an expression of $\boldsymbol{\psi}_\Delta(\boldsymbol{x}, \boldsymbol{\Gamma}(t))$ in terms of $\boldsymbol{x}$ and $\boldsymbol{\Gamma}$ has been learned, since it does not vary with the expression of γ in terms of t (see Theorem 2), it is applicable to an entire *family* of non-autonomous systems Eq. (4) where the expression of $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ in terms of $\boldsymbol{x}$ and γ is the same but the expressions of γ in terms of t differ. In contrast, the expression of $\boldsymbol{\xi}_\Delta$ is "married to" a specific choice of γ and is only applicable to a specific non-autonomous system Eq. (4) (see Theorem 3).

— Theorem 2 assumes that γ is a polynomial of t. This is not necessary in Theorem 3.

In Sections 3.3 and 3.4, we will compare the numerical results of the NN models built for both unsteady Δ-increments.

We conclude Section 2.1 with a summary of the three variants of Δ-increment described above. See Table 1. The steady Δ-increment $\boldsymbol{\phi}_\Delta$ and the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ were introduced in Qin et al. (2019, 2021). To our knowledge, the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ is novel; so are the explicit, analytic expressions for all three Δ-increments in terms of the forcing terms in the underlying dynamical systems (see Appendix B).

2.2 The Neural Network Model for the Δ-Increment

In Qin et al. (2019), several NNs that resemble the residual neural network (ResNet) (He et al., 2016) were proposed to approximate the flow map $\boldsymbol{\Phi}_\Delta$ for a steady flow [see Eq. (3)]. In Qin et al. (2021), they were extended to model the flow map $\boldsymbol{\Psi}_\Delta$ for an unsteady flow [see Eq. (7)]. Our approach is very similar: we use feedforward NNs (Jain et al., 1996) to approximate all three Δ-increments described in Section 2.1 and then compose these learned increments to construct the full flow maps.

TABLE 1: The three variants of Δ-increment (d_{flo}: dimension of the flow, 2 or 3 in this work)

$\boldsymbol{\Delta}$-increment	Input	Output	Notes
$\boldsymbol{\phi}_\Delta$	$\mathbf{x}(t) \in \mathbb{R}^{d_{\text{flo}}}$	$\mathbf{x}(t+\Delta) - \mathbf{x}(t) \in \mathbb{R}^{d_{\text{flo}}}$	termed "steady Δ-increment"
$\boldsymbol{\psi}_\Delta$	$\begin{bmatrix} \mathbf{x}(t) \\ \boldsymbol{\Gamma}(t) \end{bmatrix} \in \mathbb{R}^{d_{\text{flo}}+m+1}$		termed "γ-explicit unsteady Δ-increment" $\boldsymbol{\Gamma}(t) = \begin{bmatrix} \gamma(t) \\ \gamma^{(1)}(t) \\ \vdots \\ \gamma^{(m)}(t) \end{bmatrix}$ γ: an mth-degree polynomial
$\boldsymbol{\xi}_\Delta$	$\begin{bmatrix} \mathbf{x}(t) \\ t \end{bmatrix} \in \mathbb{R}^{d_{\text{flo}}+1}$		termed "γ-implicit unsteady Δ-increment"

Let d_{in} denote the dimension of the domain of a Δ-increment; that is, $d_{\text{in}} = d_{\text{flo}}$, $d_{\text{flo}}+m+1$, and $d_{\text{flo}}+1$ for $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, and $\boldsymbol{\xi}_\Delta$, respectively (see Table 1). For each of the three Δ-increments, their feedforward NN model is a function from $\mathbb{R}^{d_{\text{in}}}$ to $\mathbb{R}^{d_{\text{flo}}}$ that takes the following form:

$$\mathbf{N}_{\boldsymbol{\Theta}}(\boldsymbol{\mathcal{X}}) = \sigma_{N_{\text{lay}}+1}\big(W_{N_{\text{lay}}+1}(\cdots \sigma_2(W_2\sigma_1(W_1\boldsymbol{\mathcal{X}} + \mathbf{b}_1) + \mathbf{b}_2)\cdots) + \mathbf{b}_{N_{\text{lay}}+1}\big), \qquad (13)$$

where W_i is a weight matrix, $\mathbf{b}_i$ is a bias vector, σ_i is an activation function applied element-wise, N_{lay} is the number of hidden layers, and $\boldsymbol{\Theta}$ is a vector containing all the entries in the weight matrices and bias vectors, that is, $\boldsymbol{\Theta}$ is a vector of model parameters. The sizes of W_i and $\mathbf{b}_i$ depend on the number of "neurons" on the two network layers connected by them. Depending on whether the flow is steady or unsteady, the input, $\boldsymbol{\mathcal{X}}$, in Eq. (13) is either a particle's position at a time point t, or this position augmented by some extra term(s) dependent on t (see Table 1). If the model parameters in $\boldsymbol{\Theta}$ are properly chosen, then $\mathbf{N}_{\boldsymbol{\Theta}}(\boldsymbol{\mathcal{X}})$ is approximately the change in the particle's position after a time interval of length Δ. Replacing $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, or $\boldsymbol{\xi}_\Delta$ in Eqs. (3), (7), or (12) with $\mathbf{N}_{\boldsymbol{\Theta}}$, we obtain a model for the flow map $\boldsymbol{\Phi}_\Delta$, $\boldsymbol{\Psi}_\Delta$, or $\boldsymbol{\Xi}_\Delta$, which allows us to simulate the position of the particle at time $t+\Delta$.

Furthermore, we can trace a particle over the time domain $[0, T]$, where T is an integer multiple of Δ, by applying the model for the flow map recursively, as outlined in Algorithm 1.

2.3 The Training and Testing of Neural Networks

We choose the parameters in $\boldsymbol{\Theta}$ of an NN model (13) to "best fit" a training set obtained from a collection of trajectories of many particles over the time period $[0, T]$. More specifically, it

Algorithm 1: Tracing a particle using an NN model for $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, or $\boldsymbol{\xi}_\Delta$

Input: the NN model $\mathbf{N}_{\boldsymbol{\Theta}}$ for $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, or $\boldsymbol{\xi}_\Delta$,
the time duration $\Delta > 0$,
the particle's position, $\mathbf{x}^0$, at $t = 0$,
a positive integer K,
a polynomial γ of degree m (in the case of $\boldsymbol{\psi}_\Delta$ only)

Output: the estimated position of the particle, $\widetilde{\mathbf{x}}^{j\Delta}$, at $t = j\Delta$ for $j = 1,\ 2, \cdots, K$

$\widetilde{\mathbf{x}}^0 \leftarrow \mathbf{x}^0$; `// initialization`
for $j \leftarrow 1$ *to* K **do**
  switch Δ-*increment* **do**
    case $\boldsymbol{\phi}_\Delta$ **do**
      `// input for the steady` Δ`-increment`
      $\boldsymbol{\mathcal{X}} \leftarrow \widetilde{\mathbf{x}}^{(j-1)\Delta}$;
    end
    case $\boldsymbol{\psi}_\Delta$ **do**
      `// input for the` γ`-explicit unsteady` Δ`-increment`
      $\boldsymbol{\mathcal{X}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{x}}^{(j-1)\Delta} \\ \boldsymbol{\Gamma}((j-1)\Delta) \end{bmatrix}$;
    end
    case $\boldsymbol{\xi}_\Delta$ **do**
      `// input for the` γ`-implicit unsteady` Δ`-increment`
      $\boldsymbol{\mathcal{X}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{x}}^{(j-1)\Delta} \\ (j-1)\Delta \end{bmatrix}$;
    end
  end
  $\widetilde{\mathbf{x}}^{j\Delta} \leftarrow \widetilde{\mathbf{x}}^{(j-1)\Delta} + \mathbf{N}_{\boldsymbol{\Theta}}(\boldsymbol{\mathcal{X}})$; `// update the particle's position`
end

consists of N_{train} input-output pairs in the form of $\left(\boldsymbol{\mathcal{X}}_i^{t_j}, \mathbf{x}_i^{t_j+\Delta} - \mathbf{x}_i^{t_j}\right)$, where $\mathbf{x}_i^{t_j}$ is the position of the ith particle at a time point $t_j \in [0, T-\Delta]$, $\mathbf{x}_i^{t_j+\Delta}$ is its position after a time duration of length Δ, and

$$\boldsymbol{\mathcal{X}}_i^{t_j} = \begin{cases} \mathbf{x}_i^{t_j}, \text{ if the NN model is for } \boldsymbol{\phi}_\Delta; \\ \begin{bmatrix} \mathbf{x}_i^{t_j} \\ \boldsymbol{\Gamma}(t_j) \end{bmatrix}, \text{ if the NN model is for } \boldsymbol{\psi}_\Delta; \\ \begin{bmatrix} \mathbf{x}_i^{t_j} \\ t_j \end{bmatrix}, \text{ if the NN model is for } \boldsymbol{\xi}_\Delta, \end{cases} \tag{14}$$

(see Table 1). We emphasize that training an NN model for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_{\boldsymbol{\Delta}}$ requires knowing the time-dependent term γ in the fluid velocity vector $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ or a polynomial approximation to it. This is not necessary for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_{\boldsymbol{\Delta}}$, which only entails trajectory data and their time stamps. We relabel each data pair in the

training set as $\left(\boldsymbol{\mathcal{X}}_k^{\text{train}}, \mathbf{y}_k^{\text{train}} - \mathbf{x}_k^{\text{train}}\right)$ such that for each index k, there exist a particle index i and a time point t_j satisfying $\boldsymbol{\mathcal{X}}_k^{\text{train}} = \boldsymbol{\mathcal{X}}_i^{t_j}$, $\mathbf{x}_k^{\text{train}} = \mathbf{x}_i^{t_j}$, and $\mathbf{y}_k^{\text{train}} = \mathbf{x}_i^{t_j+\Delta}$. The training set can thus be denoted by

$$\left\{\left(\boldsymbol{\mathcal{X}}_k^{\text{train}}, \mathbf{y}_k^{\text{train}} - \mathbf{x}_k^{\text{train}}\right)\right\}_{k=1}^{N_{\text{train}}}. \tag{15}$$

The optimal parameter vector, $\boldsymbol{\Theta}^*$, for the NN model is the minimizer of the mean-squared error (MSE) associated with the training set

$$MSE_{\text{train}} = \frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \left\| \mathbf{N}_{\boldsymbol{\Theta}}\left(\boldsymbol{\mathcal{X}}_k^{\text{train}}\right) - \left(\mathbf{y}_k^{\text{train}} - \mathbf{x}_k^{\text{train}}\right) \right\|_2^2, \tag{16}$$

found by a numerical method.

A test set consisting of input-output pairs in the same format is also obtained from the same collection of trajectories, which does not overlap with the training set and does not enter the optimization of the model parameters in $\boldsymbol{\Theta}$. The trained NN model, $\mathbf{N}_{\boldsymbol{\Theta}^*}$, is tested on this set to examine its adaptability to new, unseen data. It is denoted by

$$\left\{\left(\boldsymbol{\mathcal{X}}_k^{\text{test}}, \mathbf{y}_k^{\text{test}} - \mathbf{x}_k^{\text{test}}\right)\right\}_{k=1}^{N_{\text{test}}}. \tag{17}$$

On a different collection of trajectories that does not overlap with the one from which data sets (15) and (17) are drawn, we also examine the accuracy of Algorithm 1, that is, the recursive application of the already-trained NN model $\mathbf{N}_{\boldsymbol{\Theta}^*}$ at predicting complete trajectories spanning over the time domain $[0, T]$, instead of trajectory segments over a time interval of length Δ.

3. NUMERICAL EXAMPLES

We examine the performance of the data-driven approach outlined in Section 2 at approximating the Δ-increments for four benchmark problems: Hill's spherical vortex (Section 3.1), the classic steady ABC flow (Section 3.2), a double-gyre flow (Section 3.3), and the unsteady ABC-type flow (Section 3.4). They consist of three three-dimensional (3D) flows, one two-dimensional (2D) flow, two steady flows, and two unsteady flows (see Table 2 for a summary), which exhibit complex behaviors such as eddies and chaotic advection, a hallmark of full turbulence. In all four problems, the velocity field is known analytically, allowing for convenient acquisition of training and test data through numerical simulation. We also examine the performance of Algorithm 1, in which the Δ-increment learned from data is applied recursively to predict fluid particle trajectories.

TABLE 2: Specifications of the benchmark problems

Example	2D/ 3D	Steady/ Unsteady	Characteristic		
			Length	Speed	Time
Hill's spherical vortex	3D	Steady	2	0.25	8
Steady ABC flow	3D	Steady	2π	$\sqrt{3}+\sqrt{2}$	2
Double-gyre flow	2D	Unsteady	1	$0.1 \cdot \pi$	4
Unsteady ABC flow	3D	Unsteady	2π	$\sqrt{3}+\sqrt{2}$	2

In each example, we follow the procedure described below to train and test a feedforward NN model for the Δ-increment $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, or $\boldsymbol{\xi}_\Delta$ (see Table 1), where $\Delta > 0$ is a duration of time. Let there be N_{lay} hidden layers in the NN and N_{neu} "neurons" on each hidden layer. In Eq. (13), the activation function σ_i is chosen to be the hyperbolic tangent function if $1 \le i \le N_{\text{lay}}$ and a linear function if $i = N_{\text{lay}} + 1$. These choices are quite standard for regression problems.

Step 1. Use the Latin hypercube sampling (LHS) (Tang, 1993) to determine the initial positions of N_{par} particles in a computational domain D taken from the literature.

Step 2. Simulate the trajectories of these particles between time 0 and an end time $T > 0$ using the explicit four-stage Runge-Kutta method (RK4) with step size $0 < \tau < \Delta$.[‡] This step entails solving N_{par} straightforward initial-value problems since the fluid velocity $\boldsymbol{u}$ in Eq. (1) or Eq. (4) is known explicitly. The trajectories generated this way are considered the "ground truth" for comparison with the predicted results of an NN. The end time T varies from problem to problem and is calculated as a characteristic length divided by a characteristic speed, both of which are listed in Table 2 for each example and further explained in its corresponding subsection.

Step 3. On each trajectory, randomly select N_{seg} segments with the starting time and end time separated by a duration Δ. As a result, we obtain $N_{\text{total}} = N_{\text{par}} \cdot N_{\text{seg}}$ pairs of data points

$$\{(\boldsymbol{\mathcal{X}}_k, \mathbf{y}_k - \mathbf{x}_k)\}_{k=1}^{N_{\text{total}}}, \tag{18}$$

from which the Δ-increment $\boldsymbol{\phi}_\Delta$, $\boldsymbol{\psi}_\Delta$, or $\boldsymbol{\xi}_\Delta$ can be learned. As in Section 2.3, for each index k in Eq. (18), there exist a particle index i and a time point t_j in $[0, T - \Delta]$ such that $\mathbf{x}_k$ is the position of the ith particle at time t_j, and $\mathbf{y}_k$ is its position at time $t_j + \Delta$; in addition, $\boldsymbol{\mathcal{X}}_k$ is as defined in Eq. (14).

Steps 1–3 constitute the process of data acquisition for the training and testing of an NN model for the Δ-increment. It is also illustrated graphically in Fig. 1 for the double-gyre example (see Section 3.3) with $\Delta = T/25 = 0.16$, $N_{\text{par}} = 10$, and $N_{\text{seg}} = 3$.

Step 4. Randomly select a training set (15) and a test set (17) from (18) that do not overlap. The ratio between their sizes, $N_{\text{train}} : N_{\text{test}}$, is about 5:1.

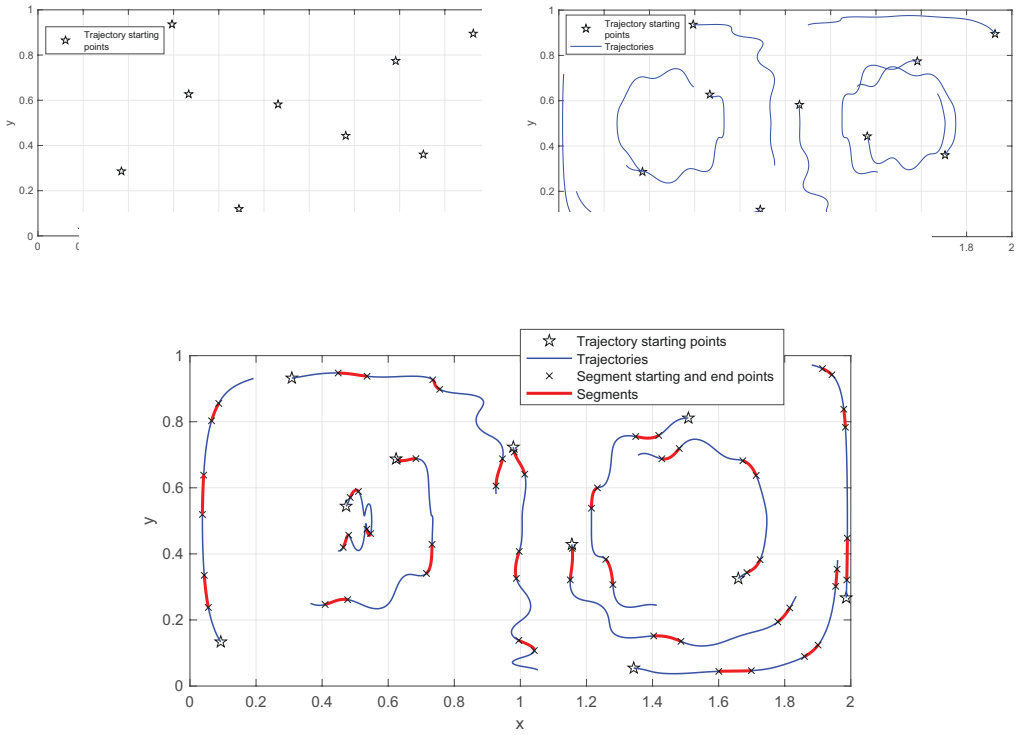
Step 5. Determine the optimal parameters of the NN model by minimizing the MSE of the training set defined in (16). We solve the minimization problem using 1000 epochs of the Levenberg-Marquardt algorithm (Marquardt, 1963) implemented in MATLAB's Deep Learning Toolbox. Let $\boldsymbol{\Theta}^*$ denote the vector of optimal parameter values found this way.

Step 6. Assess the accuracy of the trained NN model $\mathbf{N}_{\boldsymbol{\Theta}^*}$ on the test set (17), which has not been used in the training of the model in any way and can thus be viewed as new, unseen data. For $k = 1, 2, \cdots, N_{\text{test}}$, we use the segment relative error, defined as follows, to measure the accuracy of the NN model:

$$\mathcal{E}_k^{\text{seg}} = \frac{\left\| \mathbf{N}_{\boldsymbol{\Theta}^*}(\boldsymbol{\mathcal{X}}_k^{\text{test}}) - (\mathbf{y}_k^{\text{test}} - \mathbf{x}_k^{\text{test}}) \right\|_2}{\text{length of the } k\text{th segment}}, \tag{19}$$

the numerator of which is the absolute error in the estimated kth increment, $\mathbf{y}_k^{\text{test}} - \mathbf{x}_k^{\text{test}}$, from the test set (17). We note that this is different from the error (16) used to determine

[‡]We assume that T is an integer multiple of Δ, and Δ is an integer multiple of τ.



(c)

FIG. 1: Illustration of the process of data acquisition for the training and testing of an NN model for the Δ-increment. The double-gyre flow is considered. $\Delta = T/25 = 0.16$, $N_{\text{par}} = 10$, and $N_{\text{seg}} = 3$. (a) Step 1. (b) Step 2. (c) Step 3.

$\boldsymbol{\Theta}^*$ in Step 5. It allows for a fair comparison between two NN models built for two different values of Δ as it has been normalized by the length of the segment.

We also perform the following three steps to examine the accuracy of Algorithm 1 at calculating complete particle trajectories between time 0 and an end time T', which is greater than or equal to T, the trajectory end time used in Step 2.

Step 7. Simulate an additional test set of N_{traj} particle trajectories with end time T' following Steps 1 and 2. We select a different seed for the random number generator in the LHS to avoid reusing a trajectory that has already been sampled to learn the Δ-increment. Assume that T' is an integer multiple of Δ. For $i = 1, 2, \cdots, N_{\text{traj}}$ and $j = 0, 1, 2, \cdots, K$, where $K = T'/\Delta$, let $\mathbf{x}_i^{j\Delta}$ denote the point on the ith trajectory corresponding to time $j\Delta$.

Step 8. Estimate each trajectory in this set using Algorithm 1. Let $\widetilde{\mathbf{x}}_i^{j\Delta}$ denote the point on the ith estimated trajectory corresponding to time $j\Delta$. That is,

$$\begin{aligned} \widetilde{\mathbf{x}}_i^0 &= \mathbf{x}_i^0, \\ \widetilde{\mathbf{x}}_i^{j\Delta} &= \widetilde{\mathbf{x}}_i^{(j-1)\Delta} + \mathbf{N}_{\boldsymbol{\Theta}^*}\left(\boldsymbol{\mathcal{X}}_i^{(j-1)\Delta}\right) \quad \text{for} \quad 1 \le j \le K, \end{aligned} \tag{20}$$

where $\boldsymbol{\mathcal{X}}_i^{(j-1)\Delta}$ is as defined in Eq. (14) with t_j and $\mathbf{x}_i^{t_j}$ replaced by $(j-1)\Delta$ and $\widetilde{\mathbf{x}}_i^{(j-1)\Delta}$.

Step 9. Assess the accuracy of Algorithm 1 at estimating the N_{traj} trajectories. For the ith trajectory, we measure the following relative error:

$$\mathcal{E}_i^{\text{traj}} = \frac{\Delta \cdot \sum_{j=1}^{K} \left\| \widetilde{\mathbf{x}}_i^{j\Delta} - \mathbf{x}_i^{j\Delta} \right\|_2}{\text{length of the } i\text{th trajectory}}, \tag{21}$$

the numerator of which is approximately the total absolute error along the ith estimated trajectory. It allows us to compare the performance of Algorithm 1 at estimating trajectories with different end times since it has been normalized by the length of the trajectory.

A brief description of some of the parameters and their values used in the numerical experiments can be found in Table 3.

We will explore the performance of various versions of Algorithm 1. They may use NNs with different network structures and/or corresponding to different variants of Δ-increment. They may also be applied to estimate trajectories with different end times. To distinguish them from one another, we refer to each of them as Algorithm 1 (Δ-increment, Δ, N_{lay}, N_{neu}, T'). For example, Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, 0.08, 5, 20, 8) refers to the version of Algorithm 1 that calculates trajectories over the time domain $[0, 8]$ by recursively applying an NN with five hidden layers, 20 neurons on each layer built for the steady Δ-increment $\boldsymbol{\Phi}_\Delta$ where $\Delta = 0.08$.

3.1 Hill's Spherical Vortex

Hill's spherical vortex (Rockwood et al., 2019) is a classic example of a 3D vortex flow. It describes a swirling motion of a fluid vortex ring that fills a spherical volume, and the analytically defined Hill's spherical vortex has a separatrix at the boundary: fluid does not cross the spherical

TABLE 3: Summary of some parameters used in the numerical experiments

Parameter	Description	Value
Δ	Time duration between the starting and end time points of each trajectory segment	Between $T/400$ and $T/25$
τ	Step size used in RK4 to simulate particle trajectories	0.001
N_{lay}	Number of hidden layers in the NN	Between 1 and 5
N_{neu}	Number of "neurons" on each hidden layer	Between 20 and 735
N_{param}	Number of NN parameters in weight matrices and bias vectors combined, that is, in $\boldsymbol{\Theta}$	Between 1823 and 4442
N_{par}	Number of particle trajectories from which the trajectory segments are sampled	5000
N_{seg}	Number of segments sampled from each trajectory	10
N_{total}	$N_{\text{par}} \cdot N_{\text{seg}}$	50,000
N_{traj}	Number of additional particle trajectories for testing Algorithm 1	1000

vortex boundary. The velocity of the fluid is proportional to the distance from the center of the vortex ring, and the vorticity (i.e., the rotation of the fluid) is proportional to the inverse of the distance squared. Hill's spherical vortex is an important solution in fluid dynamics because it provides a simple and idealized model for the motion of a fluid with rotational symmetry. It has been used to study a wide range of phenomena, including the formation of hurricanes and other atmospheric vortices, the motion of planets and stars, and the behavior of viscous fluids.

In our example, the mathematical expression of the velocity vector field $\boldsymbol{u}(\boldsymbol{x})$ within and around Hill's spherical vortex can be represented piecewise. Let $\boldsymbol{u} = [u \;\; v \;\; w]^T$ and $\boldsymbol{x} = [x \;\; y \;\; z]^T$. Inside the spherical vortex ring boundary, the velocity components are as follows:

$$u(\boldsymbol{x}) = \frac{\alpha x z}{5}, \tag{22}$$

$$v(\boldsymbol{x}) = \frac{\alpha y z}{5}, \tag{23}$$

$$w(\boldsymbol{x}) = \frac{\alpha(r^2 - z^2 - 2x^2 - 2y^2)}{5}, \tag{24}$$

while outside of the spherical vortex, the velocity components are as follows:

$$u(\boldsymbol{x}) = \frac{\alpha r^5 x z}{5(x^2 + y^2 + z^2)^{(5/2)}}, \tag{25}$$

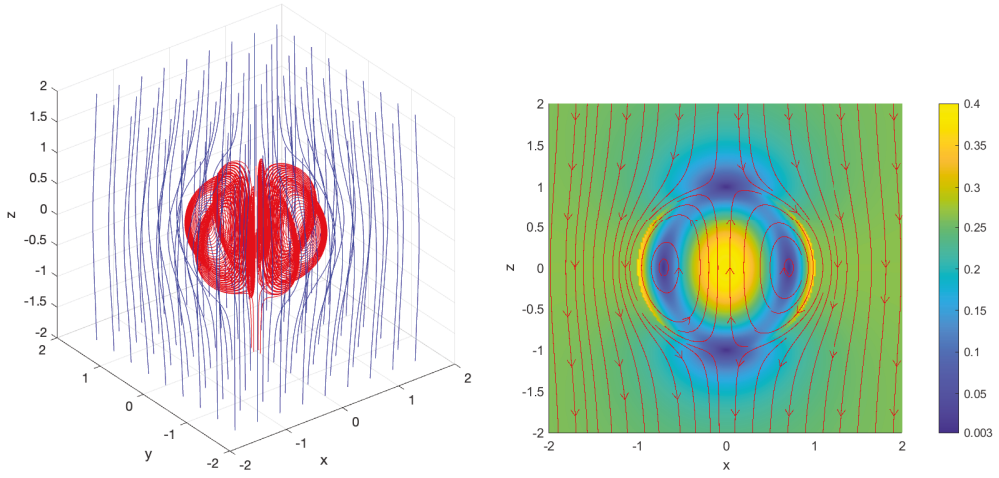
$$v(\boldsymbol{x}) = \frac{\alpha r^5 y z}{5(x^2 + y^2 + z^2)^{(5/2)}}, \tag{26}$$

$$w(\boldsymbol{x}) = \frac{-\alpha r^2\{[2(x^2 + y^2 + z^2)^{(5/2)}] - 2r^3 z^2 + r^3 y^2\}}{15(x^2 + y^2 + z^2)^{(5/2)}}. \tag{27}$$

The vortex strength parameter is $\alpha = 2$, and the radius of the sphere is $r = 1$. 3D and 2D streamlines are shown in Fig. 2.

In this example, the characteristic length is chosen to be 2, the non-dimensional diameter of the spherical vortex, and the characteristic speed is chosen to be the non-dimensional free-stream speed, 0.25, as seen in Fig. 2(b). It follows that the characteristic time is $T = 2/0.25 = 8$. The domain D from which the trajectory starting points in Steps 1 and 7 are sampled is the cube $[-2, 2] \times [-2, 2] \times [-2, 2]$. Recall that for a steady flow, the Δ-increment $\boldsymbol{\Phi}_\Delta$, defined in Eqs. (2) and (B.1), is a function from $\mathbb{R}^{d_{\text{flo}}}$ to $\mathbb{R}^{d_{\text{flo}}}$ which, given a starting position of a particle in this flow, produces the change in the particle's position after a time duration of fixed length Δ. In this example, $d_{\text{flo}} = 3$ as the flow is 3D.

We first fix $\Delta = T/100$ and examine the effectiveness of four NNs of different structures at approximating $\boldsymbol{\Phi}_\Delta$. For a fair comparison, as the number of hidden layers, N_{lay}, increases, the number of neurons on each hidden layer, N_{neu}, are adjusted accordingly so that the total number of network parameters, N_{param}, to be determined is about 2000 for every NN. We follow Steps 1–6 to construct each NN model. In Table 4, we report the minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ defined in Eq. (19) for the test set. [For a fair comparison, the data set (18) and its partition in Steps 3 and 4 are kept the same in the training and testing of NN models of different structures.] More detailed statistics of $\mathcal{E}_k^{\text{seg}}$, including its median and quartiles, are also displayed in Fig. 3(a) by box plots. From Table 4 and Fig. 3(a), we observe that as the network structure varies, the mean/median of $\mathcal{E}_k^{\text{seg}}$ can differ by as much as two orders



(a) (b)

FIG. 2: Streamlines in the Hill's spherical vortex example. (a) 3D streamlines. The red streamlines are initiated inside of the unit sphere centered at the origin. The blue streamlines are initiated on the plane $z = 2$. (b) 2D streamlines on the plane $y = 0$ calculated using the x- and z-directional velocities on the plane. The background coloring indicates the fluid speed on the plane calculated using all three velocity components. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

TABLE 4: The minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the steady Δ-increment $\boldsymbol{\Phi}_\Delta$ in the Hill's spherical vortex example. $\Delta = T/100 = 0.08$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu} and N_{param})

N_{lay}	N_{neu}	N_{param}	Min. $\mathcal{E}_k^{\text{seg}}$	Mean $\mathcal{E}_k^{\text{seg}}$	Max. $\mathcal{E}_k^{\text{seg}}$
1	300	2103	$7.37 \cdot 10^{-4}$	$2.46 \cdot 10^{-2}$	$6.98 \cdot 10^{-1}$
2	40	1923	$1.05 \cdot 10^{-5}$	$7.79 \cdot 10^{-4}$	$4.12 \cdot 10^{-2}$
3	30	2073	$3.89 \cdot 10^{-6}$	$3.22 \cdot 10^{-4}$	$3.44 \cdot 10^{-2}$
5	20	1823	$2.65 \cdot 10^{-6}$	$1.15 \cdot 10^{-4}$	$1.42 \cdot 10^{-2}$

of magnitude, indicating that network structure plays an important role in the accuracy of an NN model for $\boldsymbol{\Phi}_\Delta$. Furthermore, the deepest NN equipped with five hidden layers approximates $\boldsymbol{\Phi}_\Delta$ most accurately, despite having the fewest parameters; it boasts a mean/median $\mathcal{E}_k^{\text{seg}}$ in the order of 10^{-4}.

We also observe that prominent flow features, such as the vortex in this case, can have strong implications on the accuracy of NN models. For the NN model with $\Delta = T/100$, $N_{\text{lay}} = 5$, and $N_{\text{neu}} = 20$, in Fig. 4, we plot $\mathcal{E}_k^{\text{seg}}$ against $\|\mathbf{x}_k^{\text{test}}\|_2$, which is the distance between the starting point of the kth segment in the test set (17) and the origin. For reference, a vertical dotted line is included in this figure to mark where $\|\mathbf{x}_k^{\text{test}}\|_2$ would be exactly 1 such that $\|\mathbf{x}_k^{\text{test}}\|_2 < 1$ (inside

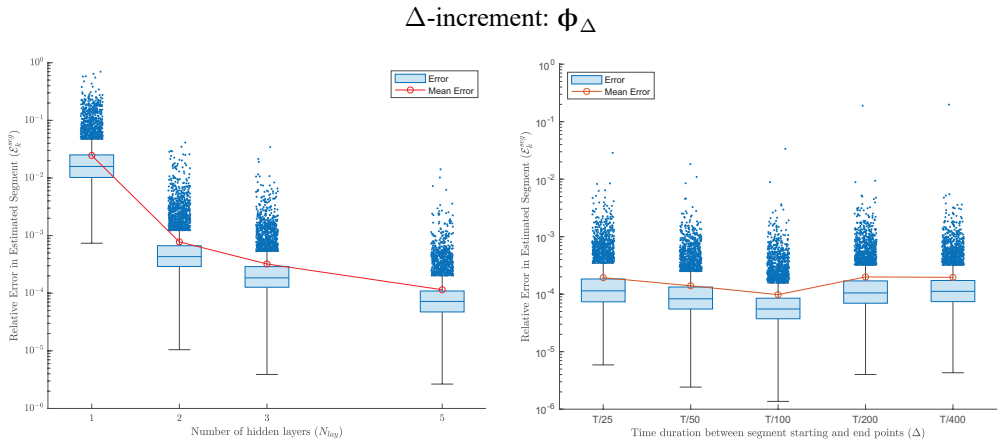


FIG. 3: Box plots of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) in the Hill's spherical vortex example. A log scale is used on the vertical axis. In each box plot, the mean (circle) and median (horizontal line through the box), first and third quartiles (lower and upper edges of the box), outliers (dots), and minimum and maximum of the nonoutliers (boundaries of the lower and upper whiskers) of $\mathcal{E}_k^{\text{seg}}$ are shown. Outliers are defined to be $\mathcal{E}_k^{\text{seg}}$ that are 1.5 times of the interquartile range (height of the box) below the first quartile or above the third quartile. (a) The NN models for the steady Δ-increment Φ_Δ with fixed $\Delta = T/100 = 0.08$ and four different network structures (see Table 4). (b) The NNs with fixed network structure ($N_{\text{lay}} = 5$, $N_{\text{neu}} = 20$) and five different values of Δ ($T/25 = 0.32$, $T/50 = 0.16$, $T/100 = 0.08$, $T/200 = 0.04$, and $T/400 = 0.02$).

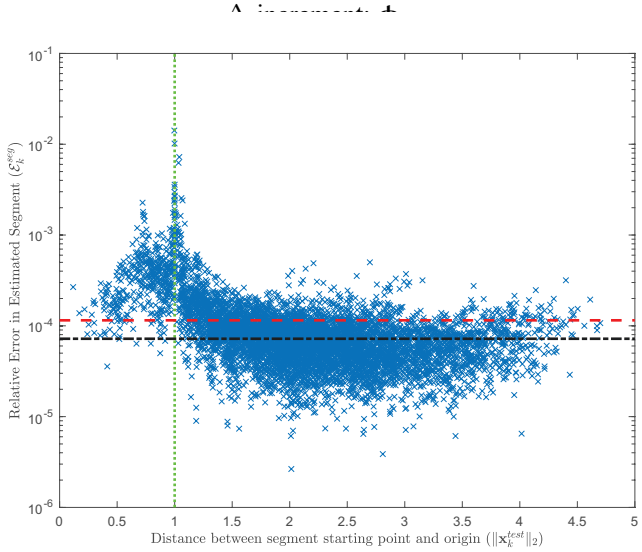


FIG. 4: The segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) against the distance between the segment starting point $\boldsymbol{x}_k^{\text{test}}$ and the origin in the Hill's spherical vortex example. A semi-log scale is used. $\Delta = T/100 = 0.08$, $N_{\text{lay}} = 5$, $N_{\text{neu}} = 20$. The vertical dotted line marks the boundaryof the vortex. The horizontal dashed line marks the mean $\mathcal{E}_k^{\text{seg}}$. The horizontal dashed-dotted line marks the median of $\mathcal{E}_k^{\text{seg}}$.

the spherical vortex) to its left, and $\|\mathbf{x}_k^{\text{test}}\|_2 > 1$ (outside the spherical vortex) to its right. A horizontal dashed line and a horizontal dashed-dotted line are also included to mark the mean and median $\mathcal{E}_k^{\text{seg}}$, respectively. As this figure indicates, for most cases where $\|\mathbf{x}_k^{\text{test}}\|_2 \approx 1$ or $\|\mathbf{x}_k^{\text{test}}\|_2 < 1$, $\mathcal{E}_k^{\text{seg}}$ is larger than its mean and median; in particular, the largest $\mathcal{E}_k^{\text{seg}}$ occurs when $\|\mathbf{x}_k^{\text{test}}\|_2 \approx 1$, that is, when $\mathbf{x}_k^{\text{test}}$ is close to the unit sphere, the boundary of the vortex. As shown in Fig. 2, the flow pattern undergoes drastic changes near the boundary and is far more complex on the inside.

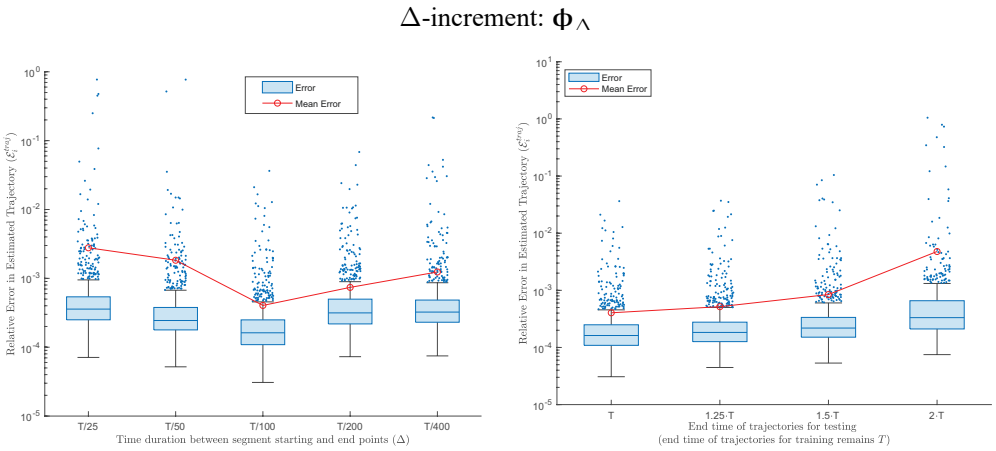
Next, we examine how the choice of Δ affects the accuracy of NN models for $\boldsymbol{\Phi}_\Delta$ by fixing $N_{\text{lay}} = 5$, $N_{\text{neu}} = 20$ and varying Δ between $T/400 = 0.02$ and $T/25 = 0.32$. For each value of Δ, we again follow Steps 1–6 to build the NN. (For a fair comparison, the trajectories in Step 2, once generated, are kept the same in the training and testing of NN models for different values of Δ.) The statistics of $\mathcal{E}_k^{\text{seg}}$ are illustrated by box plots in Fig. 3(b). We observe that the mean/median $\mathcal{E}_k^{\text{seg}}$ is consistently on the order of 10^{-4} and not very sensitive to the choice of Δ; $\mathcal{E}_k^{\text{seg}}$ achieves the minimum when Δ is neither too small nor too large, at $T/100$.

All the experiments so far aim to inspect the performance of NN models at estimating the increment in a fluid particle's position over a relatively small time duration of length Δ. We now turn to the performance of Algorithm 1, where a trained NN model is applied recursively to estimate the complete trajectory of the particle over the time domain $[0, T']$, where $T' \geq T$. In all the experiments below, $N_{\text{lay}} = 5$, and $N_{\text{neu}} = 20$.

Recall that following Steps 1–6, we have already built five NNs with $N_{\text{lay}} = 5$, $N_{\text{neu}} = 20$ for five values of Δ between $T/400$ and $T/25$, whose performance at estimating changes in fluid particles' positions is summarized and compared in Fig. 3(b). We now follow Steps 7–9 to investigate how the choice of Δ affects the accuracy of Algorithm 1. More specifically, we first follow Step 7 to generate a new set of N_{traj} trajectories with end time T. For a fair comparison, this set is kept the same as we vary Δ in Algorithm 1. Then for each Δ, we repeat Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, Δ, 5, 20, T)[§] and depict the statistics of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ defined in (21) in Fig. 5(a). For all five values of Δ, the median of $\mathcal{E}_i^{\text{traj}}$ is in the order of 10^{-4}; and while more variability can be observed for the mean of $\mathcal{E}_i^{\text{traj}}$, it stays well below 10^{-2}. Moreover, the median/mean $\mathcal{E}_i^{\text{traj}}$ also achieves the minimum at $\Delta = T/100$, as observed for the segment relative error $\mathcal{E}_k^{\text{seg}}$ in Fig. 3(b). For this value of Δ, we plot the fifty most inaccurate estimated trajectories (circles) as well as their exact counterparts (solid lines) in Figs. 6(a) and 6(b) viewed from two different angles. All trajectories but one lie within or close to the unit sphere boundary [also shown in Figs. 6(a) and 6(b) for reference], where the NNs have the most difficulty estimating the increment in a fluid particle's position, as demonstrated in Fig. 4. Additionally, in Fig. 6(c), we zoom in on the six most inaccurate estimated trajectories and the exact ones corresponding to them. The error $\mathcal{E}_i^{\text{traj}}$ in each estimated trajectory and the unit sphere boundary are displayed as well.

In the previous set of experiments, the same trajectory end time T is used to acquire the trajectories for training NN models (Step 2) and those for testing Algorithm 1 (Step 7). Naturally, we wonder about the applicability of the NNs beyond the time point T, that is, whether and to what extent they can be used to extrapolate trajectory data. To look into this, we repeat Steps 7–9 for four end times between T and $2T$, while fixing $\Delta = T/100$, $N_{\text{lay}} = 5$, and $N_{\text{neu}} = 20$. That is, the end time in Step 2 remains T for the trajectories used to train the NN, whereas

[§]Recall that this is the version of Algorithm 1 that calculates trajectories over the time domain $[0, T]$ by recursively applying an NN with five hidden layers, 20 neurons on each layer built for the steady Δ-increment $\boldsymbol{\Phi}_\Delta$.



(a) (b)

FIG. 5: Box plots of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with Algorithm 1 in the Hill's spherical vortex example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, Δ, 5, 20, T) where Δ equals $T/25 = 0.32$, $T/50 = 0.16$, $T/100 = 0.08$, $T/200 = 0.04$, or $T/400 = 0.02$. (b) Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/100$, 5, 20, T'), where T' equals $T = 8$, $1.25T = 10$, $1.5T = 12$, or $2T = 16$.

the end time T' in Step 7 can be greater than T for the trajectories used to test Algorithm 1. (For a fair comparison, the trajectory starting points in Step 7, once sampled, remain unchanged. We extend its trajectory as T' increases.) For each end time T' considered, the statistics of $\mathcal{E}_i^{\text{traj}}$ associated with Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/100$, 5, 20, T') are summarized in a box plot in Fig. 5(b). We observe that $\mathcal{E}_i^{\text{traj}}$ increases with T'; and while the increase is slow between T and $1.5T$, it becomes considerably steeper between $1.5T$ and $2T$. When $T' = 1.5T$, the mean/median $\mathcal{E}_i^{\text{traj}}$ is still below 10^{-3} and the maximum $\mathcal{E}_i^{\text{traj}}$ is around 10^{-1}, indicating that Algorithm 1 can be applied to trace fluid particles up to time $1.5T$ to a reasonable degree of accuracy, even though the training data for the NN are all collected at time points that do not exceed T. (See Appendix C for additional graphics.)

3.2 Steady ABC Flow

The steady ABC flow (Haller, 2005) is a useful test case because it exhibits a wide range of flow phenomena, including turbulence, boundary layers, and complex flow structures. It is characterized by a 3D periodic flow pattern that is created by the motion of three vortices rotating around perpendicular axes.

In this example, the fluid velocity vector field $\boldsymbol{u}(\boldsymbol{x})$ where $\boldsymbol{u} = [u\ v\ w]^T$ and $\boldsymbol{x} = [x\ y\ z]^T$ is given by

$$u(\boldsymbol{x}) = A\sin z + C\cos y, \tag{28}$$

$$v(\boldsymbol{x}) = B\sin x + A\cos z, \tag{29}$$

$$w(\boldsymbol{x}) = C\sin y + B\cos x. \tag{30}$$

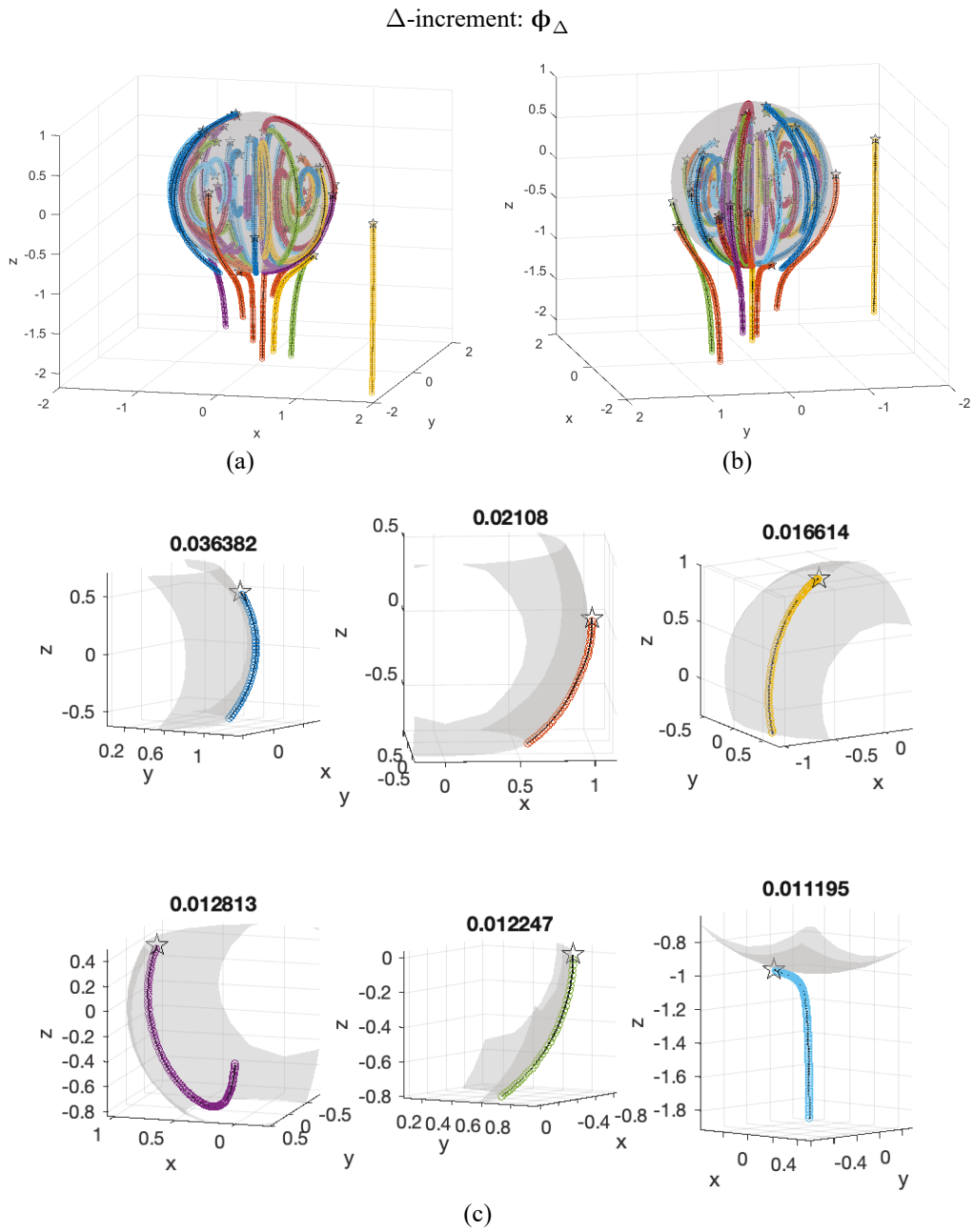


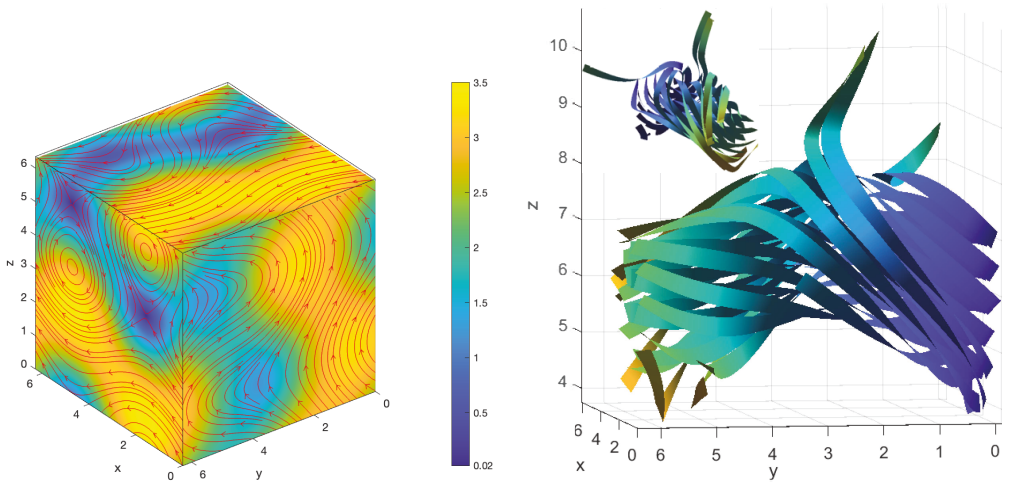
FIG. 6: The fifty most inaccurate trajectories estimated by Algorithm 1 (Φ_Δ, $T/100$, 5, 20, T) and the corresponding true trajectories in the Hill's spherical vortex example. The circles mark the estimated trajectories, and the solid lines represent the true trajectories. The starting point of each trajectory is marked with a star. (a), (b) All 50 pairs of trajectories viewed from two different angles. The unit sphere centered at the origin is included for reference. (c) The six most inaccurate estimated trajectories and the corresponding exact trajectories. The relative trajectory errors $\mathcal{E}_i^{\text{traj}}$ (21) are displayed at the top of the respective panels.

In particular, we consider the parameter values $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1$ as in Haller (2005). We show the 2D streamlines on three planes in Fig. 7(a) and a bundle of 3D "stream ribbons" in Fig. 7(b), where the twist in the ribbons is proportional to the curl of the flow.

Like the Hill's spherical vortex example, the steady ABC flow is a 3D steady flow. Therefore, we proceed as in Section 3.1 to examine the performance of NN models at advancing fluid particles' positions. In Steps 2 and 7, the particles are sampled from the domain $D = [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$. The characteristic length of the steady ABC is chosen to be 2π, the length of the edges of D. The maximum amplitude in the three velocity components, $A + B = (\sqrt{3} + \sqrt{2})$, is identified as the characteristic speed. Consequently, the characteristic time T is determined to be 2 by rounding $2\pi/(A + B)$ to the next bigger integer.

We first explore how the structure of the NN influences the performance of NN models for the steady Δ-increment $\mathbf{\Phi}_\Delta$ where $\Delta = T/100$ is fixed. We adopt the same four network structures used in Section 3.1, the number of parameters, N_{param}, of which stays around 2000 as the number of hidden layers, N_{lay}, and the number of neurons on each layer, N_{neu}, vary. The minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ defined in Eq. (19) are presented in Table 5. For each network structure, we also present more detailed statistics of $\mathcal{E}_k^{\text{seg}}$ in a box plot in Fig. 8(a). We observe that the mean and median of $\mathcal{E}_k^{\text{seg}}$ are around 10^{-4} for all four networks. Unlike in the previous example where a deeper network is more accurate, the network with a single hidden layer and the most network parameters outperforms the other networks.

We also observe that the accuracy of the NN model depends on the balance between the local rate of strain and rate of rotation, which can be characterized by the Q criterion (Hunt et al., 1988). Roughly speaking, the Q criterion at a given point increases with the rate of rotation and decreases with the rate of strain at that point. A positive Q indicates the dominance of rotation

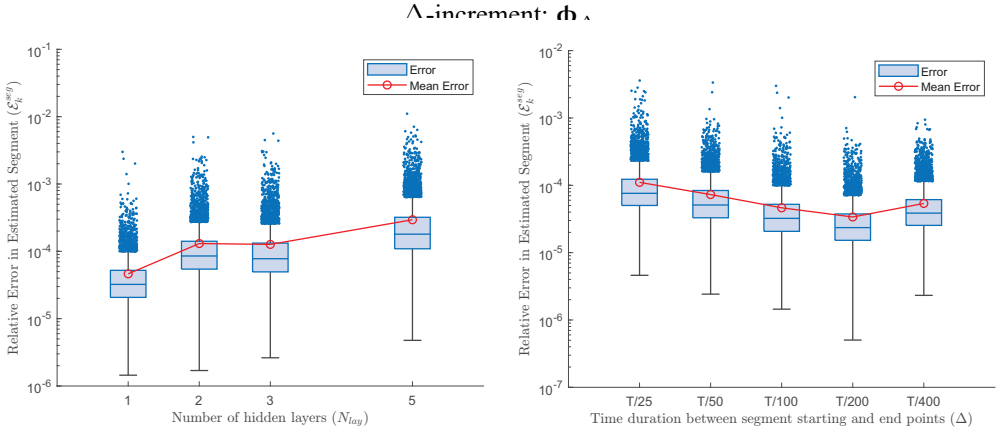


(a) (b)

FIG. 7: Streamlines and stream ribbons in the steady ABC flow example. (a) 2D streamlines calculated using the 2D fluid velocity vector field on each of the three planes $x = 0$, $y = 2\pi$, and $z = 2\pi$. The background coloring of each plane indicates the fluid speed on the plane calculated using the full 3D velocity vector field. (b) A bundle of 3D stream ribbons initiated from the plane $y = 0$. The inset shows the same bundle viewed from a different angle.

TABLE 5: The minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the steady Δ-increment $\boldsymbol{\Phi}_\Delta$ in the classic steady ABC flow example. $\Delta = T/100 = 0.02$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu}, and N_{param})

N_{lay}	N_{neu}	N_{param}	Min. $\mathcal{E}_k^{\text{seg}}$	Mean $\mathcal{E}_k^{\text{seg}}$	Max. $\mathcal{E}_k^{\text{seg}}$
1	300	2103	$1.45 \cdot 10^{-6}$	$4.63 \cdot 10^{-5}$	$3.00 \cdot 10^{-2}$
2	40	1923	$1.70 \cdot 10^{-6}$	$1.31 \cdot 10^{-4}$	$5.00 \cdot 10^{-2}$
3	30	2073	$2.63 \cdot 10^{-6}$	$1.26 \cdot 10^{-4}$	$5.62 \cdot 10^{-2}$
5	20	1823	$4.76 \cdot 10^{-6}$	$2.94 \cdot 10^{-4}$	$1.10 \cdot 10^{-2}$



(a) (b)

FIG. 8: Box plots of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) in the steady ABC flow example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) The NN models for the steady Δ-increment $\boldsymbol{\Phi}_\Delta$ with fixed $\Delta = T/100$ and four different network structures (see Table 5). (b) The NNs with fixed network structure ($N_{\text{lay}} = 1$, $N_{\text{neu}} = 30$) and five different values of Δ ($T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, $T/200 = 0.01$, and $T/400 = 0.005$).

over strain, whereas a negative Q value indicates the dominance of strain over rotation. In Fig. 9, for the NN model with $N_{\text{lay}} = 1$, $N_{\text{neu}} = 300$, and $\Delta = T/200$, we plot $\mathcal{E}_k^{\text{seg}}$ against the Q value at the starting point for every segment in the test set (17). It shows that the NN model tends to be less accurate as the initial Q decreases, that is, as the strain becomes more dominant at the segment starting location. In particular, for the majority of the segments with initial Q less than -2, $\mathcal{E}_k^{\text{seg}}$ is above the mean and median.

We next look into how the value of Δ affects the accuracy of NN models of $\boldsymbol{\Phi}_\Delta$ by fixing $N_{\text{lay}} = 1$, $N_{\text{neu}} = 300$ and varying Δ between $T/400 = 0.02$ and $T/25 = 0.32$. For each value of Δ, we again follow Steps 1–6 to build an NN. The statistics of $\mathcal{E}_k^{\text{seg}}$ are illustrated by box plots in Fig. 8(b). The mean/median $\mathcal{E}_k^{\text{seg}}$ is consistently in the order of 10^{-4}, and $\mathcal{E}_k^{\text{seg}}$ achieves the minimum at $\Delta = T/200$.

Following Steps 7–9, we also examine the performance of Algorithm 1 at estimating particle trajectories. As in Section 3.1, we perform two sets of experiments. First, we fix the network

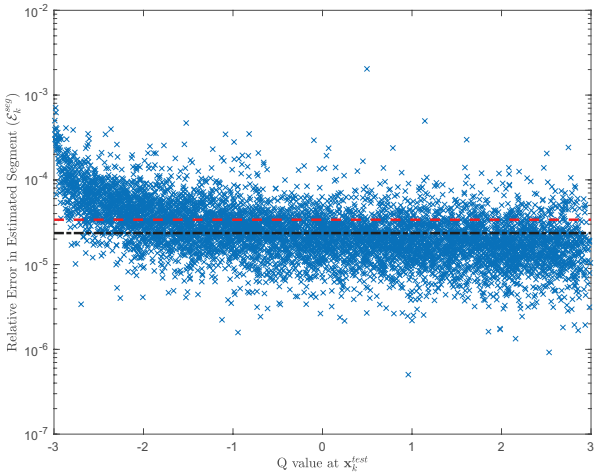
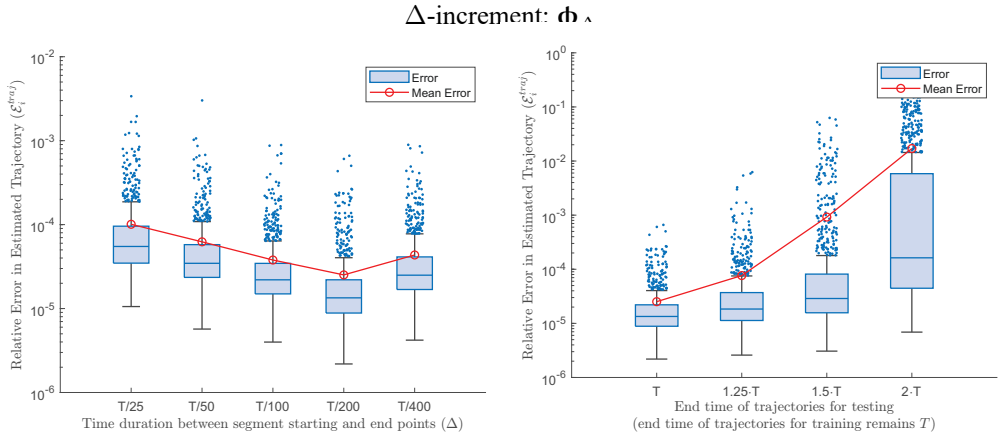


FIG. 9: The segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) against the initial Q value for the test set (17) in the steady ABC flow example. A semi-log scale is used. $\Delta = T/200 = 0.01$, $N_{\text{lay}} = 1$, $N_{\text{neu}} = 300$, and $\Delta = T/200 = 0.01$. The horizontal dashed line marks the mean $\mathcal{E}_k^{\text{seg}}$. The horizontal dash-dotted line marks the median of $\mathcal{E}_k^{\text{seg}}$.

structure ($N_{\text{lay}} = 1$, $N_{\text{neu}} = 300$) and $T' = T$ in Step 7, that is, the end time of the trajectories generated for testing Algorithm 1 is the same as the end time of the trajectories generated for training the NN in Step 2. For five values of Δ, we report the statistics of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ defined in (21) in Fig. 10(a). The mean/median $\mathcal{E}_i^{\text{traj}}$ is in the order of 10^{-4} in all cases.



(a) (b)

FIG. 10: Box plots of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with Algorithm 1 in the steady ABC flow example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, Δ, 1, 300, T) where Δ equals $T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, $T/200 = 0.01$, or $T/400 = 0.005$. (b) Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/200$, 1, 300, T') where T' equals $T = 2$, $1.25T = 2.5$, $1.5T = 3$, or $2T = 4$.

We plot the fifty least-accurate trajectories (circles) estimated by Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/200$, 1, 300, T) and the corresponding true trajectories (solid lines) in Figs. 11(a) and 11(b). The isosurfaces of Q values 2 and -2 are also shown in the two subplots. (Interestingly, each isosurface resembles a network of interconnected tubes.) We can see that most of the 50 trajectories initiate outside of the isosurface of Q value 2 [Fig. 11(a)] and within the isosurface of Q value -2 [Fig. 11(b)]. This is consistent with our observation from Fig. 9 that the NN model tends to be less accurate when strain is more dominant than rotation. Furthermore, as seen in Fig. 12, the six most inaccurate trajectories all start within the isosurface of Q value -2.

Next, we fix the NN ($N_{\text{lay}} = 1$, $N_{\text{neu}} = 300$, and $\Delta = T/200$) and repeat Steps 7–9 for four end times between $T' = T$ and $T' = 2T$. For each end time T', the statistics of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/200$, 1, 300, T') are shown in Fig. 10(b). As in the previous example, when the end time is as large as $T' = 1.5T$, the maximum $\mathcal{E}_i^{\text{traj}}$ is around 10^{-1}, indicating that the NN model can be applied to trace fluid particles up to time $T' = 1.5T$ to a reasonable degree of accuracy, even though the training data (15) for the NN are all collected at time points that do not exceed T.

3.3 Double-Gyre Flow

The double-gyre flow (Shadden et al., 2005) is a model of oceanic circulation patterns comprising two rotating eddies, or gyres, that are found in the surface layer of the ocean. These kinds of flows are created by the interaction between the wind-driven surface currents and the underlying,

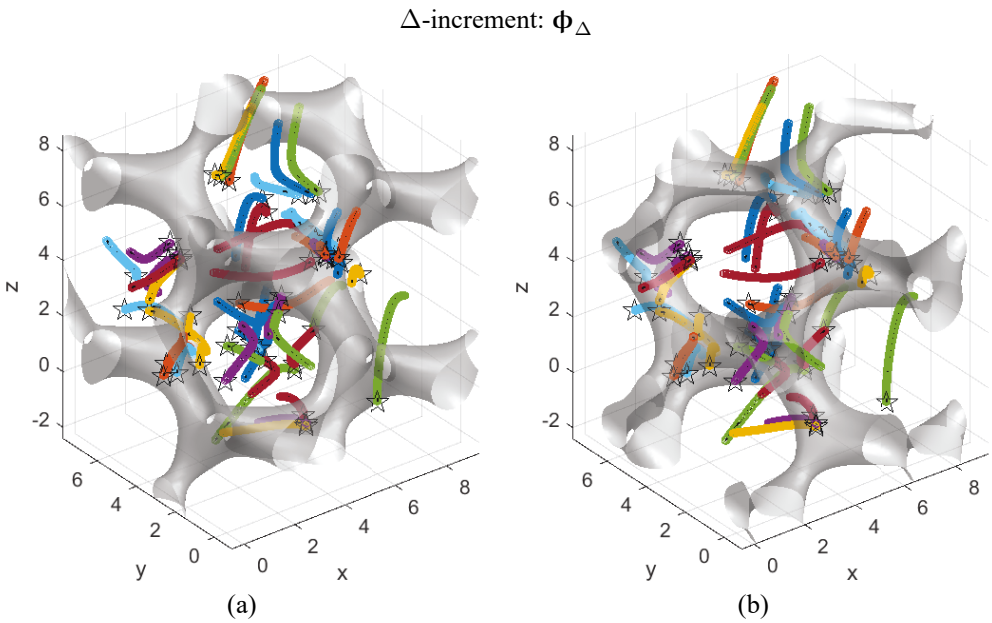


FIG. 11: The fifty least accurate trajectories estimated by Algorithm 1 ($\boldsymbol{\Phi}_\Delta$, $T/200$, 1, 300, T), the corresponding true trajectories, and Q isosurfaces in the steady ABC flow example. The circles mark the estimated trajectories, and the solid lines represent the true trajectories. The starting point of each trajectory is marked with a star. (a) The fifty pairs of trajectories shown with the isosurface of Q value 2. (b) The 50 pairs of trajectories shown with the isosurface of Q value -2.

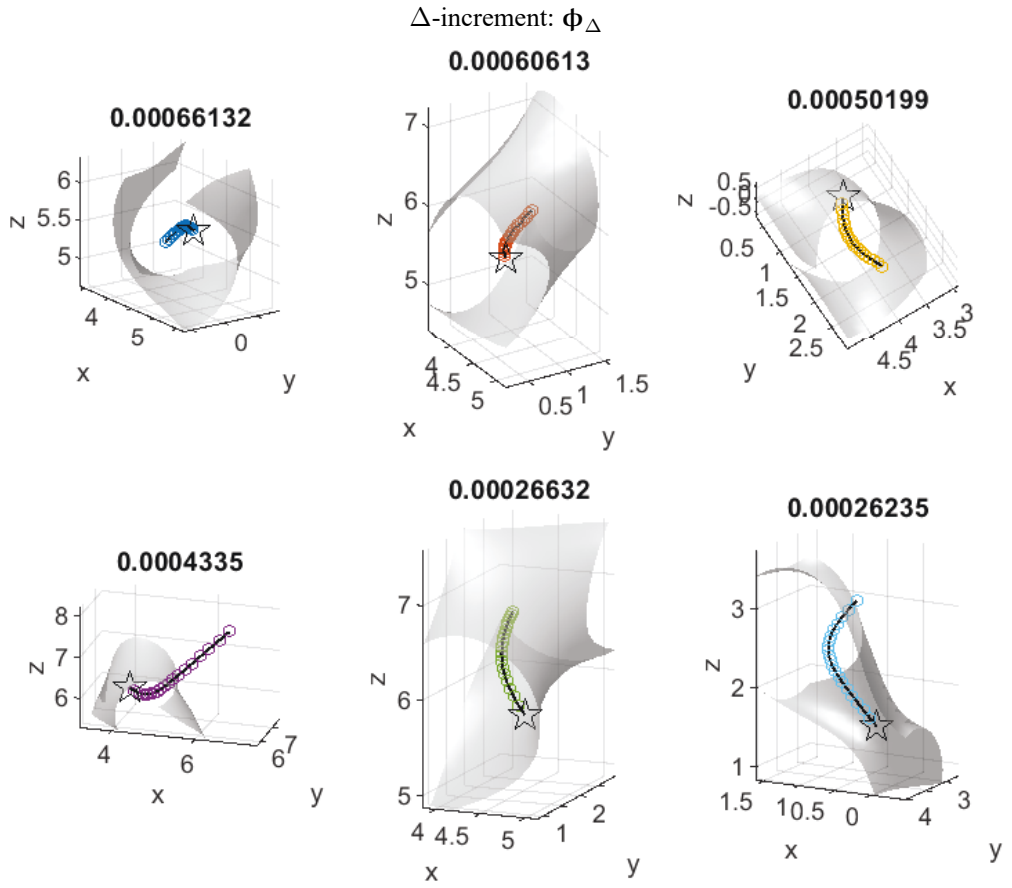


FIG. 12: The six least-accurate trajectories estimated by Algorithm 1 (Φ_Δ, $T/200$, 1, 300, T), the corresponding true trajectories, and the isosurface of Q value -2 in the steady ABC flow example. The circles mark the estimated trajectories, and the solid lines represent the true trajectories. The starting point of each trajectory is marked with a star. The trajectory relative errors $\mathcal{E}_i^{\text{traj}}$ (21) are displayed at the top of the respective panels.

stratified layers of the ocean. Researchers use the double-gyre example to study a wide range of oceanic phenomena, including the transport of nutrients and pollutants, the formation and movement of marine ecosystems, and the impact of ocean circulation patterns on climate change. By studying the behavior of the double-gyre flow, researchers can gain insights into the behavior of more complex oceanic flows and develop more accurate and efficient numerical models for simulating them.

For this 2D unsteady flow, the velocity vector field $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ is

$$u(\boldsymbol{x}, \gamma(t)) = -\pi A \sin(\pi f(x,t)) \cos(\pi y), \tag{31}$$

$$v(\boldsymbol{x}, \gamma(t)) = \pi A \cos(\pi f(x,t)) \sin(\pi y) \frac{\partial f}{\partial x}(x,t), \tag{32}$$

where $\boldsymbol{u} = [u \;\; v]^T$, $\boldsymbol{x} = [x \;\; y]^T$, and

$$f(x,t) = \epsilon \sin(\omega t) x^2 + (1 - 2\epsilon \sin(\omega t)) x. \tag{33}$$

Equations (31)–(33) imply that the time-dependent term is $\gamma(t) = \epsilon \sin(\omega t)$ in this example. We choose $A = 0.1$, $\epsilon = 0.25$, and $\omega = 2\pi$ as in Shadden et al. (2005). The flow velocity vector field when $t = 0,\ 0.25,\ 0.5,\ 0.75$ is illustrated in Fig. 13. Note that as suggested by Eqs. (32) and (33), the velocity fields are identical at $t = 0$ and 0.5 [see Figs. 13(a) and 13(c)].

The domain D from which the starting points of fluid particle trajectories are sampled in Steps 1 and 7 is $[0, 2] \times [0, 1]$. At time $t = 0$ or $t = 0.5$, the diameter of either gyre is 1 [see Figs. 13(a) and 13(c)], which is chosen to be the characteristic length. The amplitude of the right-hand sides of Eqs. (31) and (32), πA, is taken to be the characteristic speed. Rounding the ratio $1/(\pi A)$ up to the next larger integer results in the characteristic time $T = 4$. Since the flow is unsteady, we also compare the performance of NN models built for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ defined in Eqs. (5), (B.3), and the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ defined in Eqs. (11), (B.5). Recall that they are two variants resulting from two different ways of rewriting the original non-autonomous system Eq. (4) into an autonomous system: Eq. (6) or Eq. (10). We have summarized and compared them in Table 1 as well as in the text preceding it.

As seen in Table 1, $\boldsymbol{\psi}_\Delta$ is a function from $\mathbb{R}^{d_{\text{flo}}+m+1}$ to $\mathbb{R}^{d_{\text{flo}}}$, where d_{flo} is the dimension of the flow, and m is the degree of the polynomial γ of time t in the velocity vector field $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$. In this example, $d_{\text{flo}} = 2$, and $\gamma = \epsilon \sin(\omega t)$, which is not a polynomial. Therefore, for the starting time point t_k of each trajectory segment in Eq. (18) selected in Step 3, we use the second-degree Taylor polynomial of γ around t_k to approximate γ locally on that segment, between time points t_k and $t_k + \Delta$. This leads to a $\boldsymbol{\psi}_\Delta$ that is from $\mathbb{R}^5$ to $\mathbb{R}^2$. The other unsteady Δ-increment, $\boldsymbol{\xi}_\Delta$, is a function from $\mathbb{R}^3$ to $\mathbb{R}^2$ in this example, according to Table 1.

As in the previous two examples, we first follow Steps 1–6 to examine the performance of NN models for $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$, which is summarized in Tables 6 and 7 and depicted in Fig. 14. Various network structures and values of Δ are again considered. By comparing Table 6 to

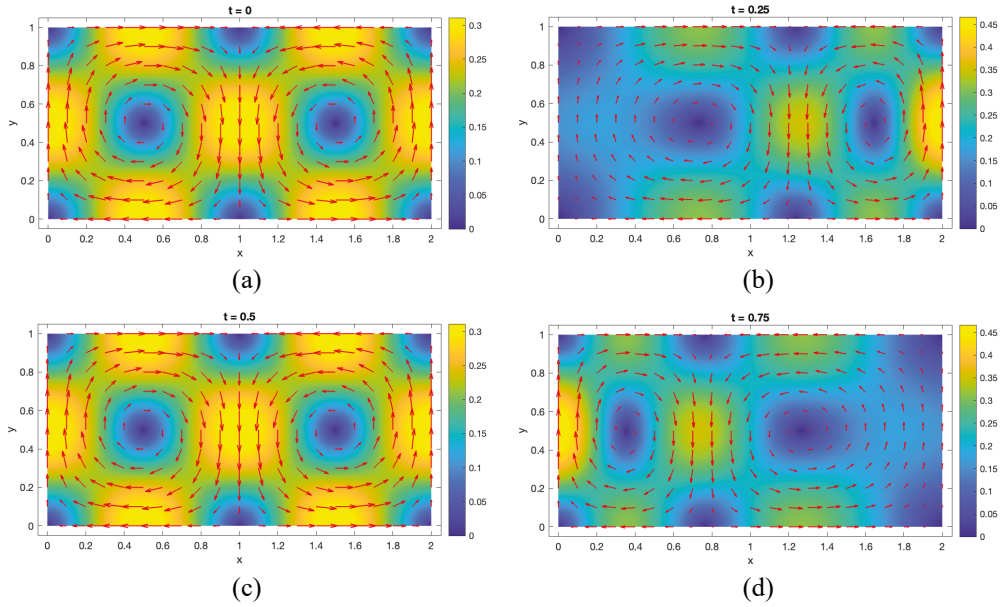


FIG. 13: Snapshots of the double-gyre flow. The lengths of the arrows and the background coloring indicate fluid speed. (a) $t = 0$. (b) $t = 0.25$. (c) $t = 0.5$. (d) $t = 0.75$.

TABLE 6: The minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ in the double-gyre example. $\Delta = T/200 = 0.02$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu} and N_{param})

$\boldsymbol{N_{\text{lay}}}$	$\boldsymbol{N_{\text{neu}}}$	$\boldsymbol{N_{\text{param}}}$	**Min.** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$	**Mean** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$	**Max.** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$
1	250	2002	$5.20 \cdot 10^{-6}$	$7.43 \cdot 10^{-4}$	$4.47 \cdot 10^{-2}$
2	40	1962	$4.76 \cdot 10^{-6}$	$7.83 \cdot 10^{-5}$	$1.22 \cdot 10^{-2}$
3	30	2102	$1.03 \cdot 10^{-7}$	$4.12 \cdot 10^{-5}$	$6.71 \cdot 10^{-3}$
4	25	2152	$2.61 \cdot 10^{-7}$	$5.97 \cdot 10^{-5}$	$3.54 \cdot 10^{-3}$

TABLE 7: The minimum, mean, and maximum of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ in the double-gyre example. $\Delta = T/200 = 0.02$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu} and N_{param})

$\boldsymbol{N_{\text{lay}}}$	$\boldsymbol{N_{\text{neu}}}$	$\boldsymbol{N_{\text{param}}}$	**Min.** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$	**Mean** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$	**Max.** $\boldsymbol{\mathcal{E}_k^{\text{seg}}}$
1	735	4412	$3.09 \cdot 10^{-5}$	$6.24 \cdot 10^{-3}$	$2.85 \cdot 10^{-1}$
2	63	4412	$1.41 \cdot 10^{-5}$	$8.14 \cdot 10^{-4}$	$1.98 \cdot 10^{-1}$
3	45	4412	$3.85 \cdot 10^{-6}$	$4.98 \cdot 10^{-4}$	$7.89 \cdot 10^{-2}$
4	37	4442	$4.56 \cdot 10^{-6}$	$4.56 \cdot 10^{-4}$	$6.78 \cdot 10^{-2}$

Table 7 and Figs. 14(a) and 14(b) to Figs. 14(c) and 14(d), we note that the NN models for $\boldsymbol{\xi}_\Delta$ are noticeably less accurate than the NN models for $\boldsymbol{\psi}_\Delta$. For every combination of Δ and N_{lay} considered, even though twice as many network parameters are used in the NN model for $\boldsymbol{\xi}_\Delta$, the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) associated with it is still about one order of magnitude larger than the $\mathcal{E}_k^{\text{seg}}$ associated with the NN model for $\boldsymbol{\psi}_\Delta$. (Note that $N_{\text{param}} \approx 4400$ in the case of $\boldsymbol{\xi}_\Delta$, and $N_{\text{param}} \approx 2000$ in the case of $\boldsymbol{\psi}_\Delta$.) In addition, as seen in Figs. 14(a) and 14(c), network structure plays an important role in the accuracy of NN models for both $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$, and deeper networks are more desirable when the total number of network parameters remains more or less the same. Figures 14(b) and 14(d) suggest that the accuracy of the NN models for both $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$ is not very sensitive to the value of Δ. For every Δ considered, the mean/median $\mathcal{E}_k^{\text{seg}}$ is in the order of 10^{-5} for the NN model for $\boldsymbol{\psi}_\Delta$ and is in the order of 10^{-4} for the NN model for $\boldsymbol{\xi}_\Delta$.

We emphasize that although the trained NN models for $\boldsymbol{\psi}_\Delta$ are more accurate, to train or simulate such a model entails knowing the time-dependent term γ in the velocity vector field $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$; that is, we need to know how the fluid velocity varies with time, which is not realistic in a setting where only trajectory data are available and the flow is simply a "black box." Learning $\boldsymbol{\xi}_\Delta$ only requires trajectory data and the time stamps at which they are collected, making it more applicable in precisely the type of scenarios where data-driven methods are highly sought after.

As in Section 3.1, we also look into the effects of distinct flow structures (the two gyres) on the accuracy of NN models for $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$. We fix $\Delta = T/200$ and $N_{\text{lay}} = 3$, $N_{\text{neu}} = 30$ for $\boldsymbol{\psi}_\Delta$, $N_{\text{lay}} = 3$, $N_{\text{neu}} = 45$ for $\boldsymbol{\xi}_\Delta$. In Fig. 15, for both NNs, we plot $\mathcal{E}_k^{\text{seg}}$ against the smaller of the following two distances: 1) $\left\|\mathbf{x}_k^{\text{test}} - [0.5 \;\; 0.5]^T\right\|_2$, that is, the distance between the starting

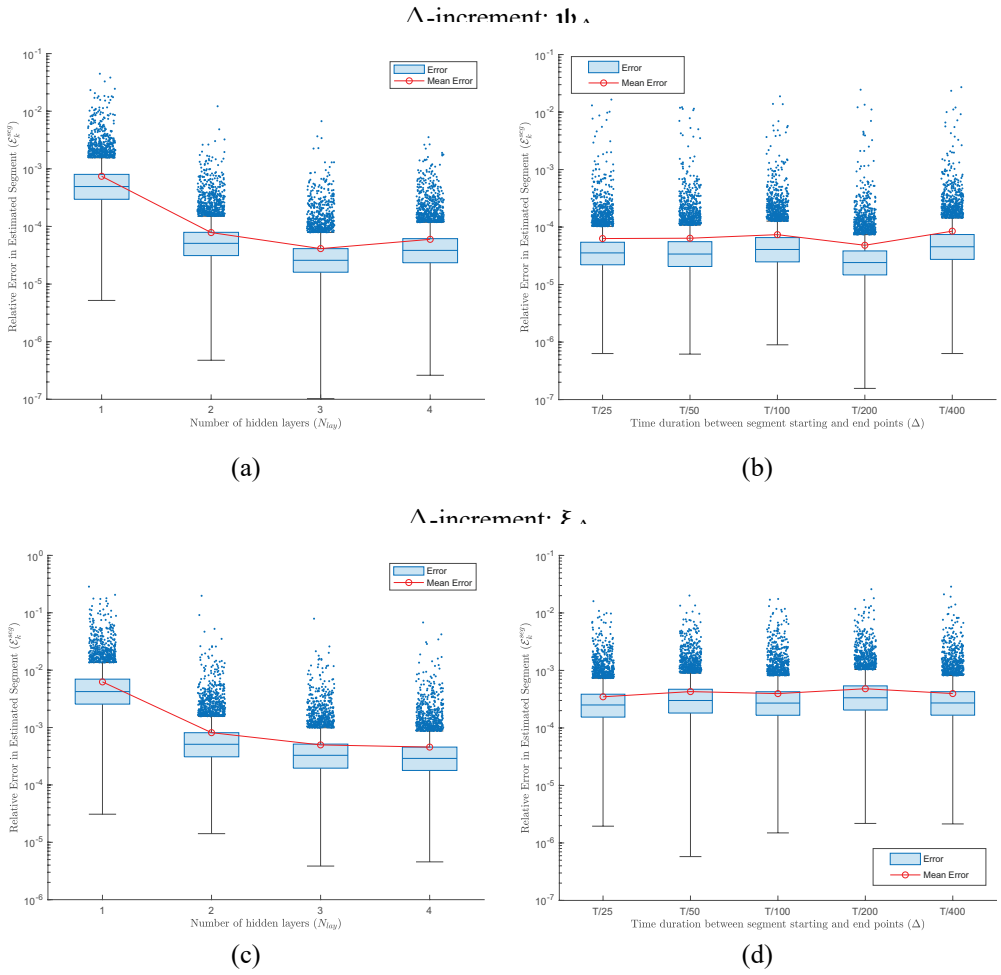
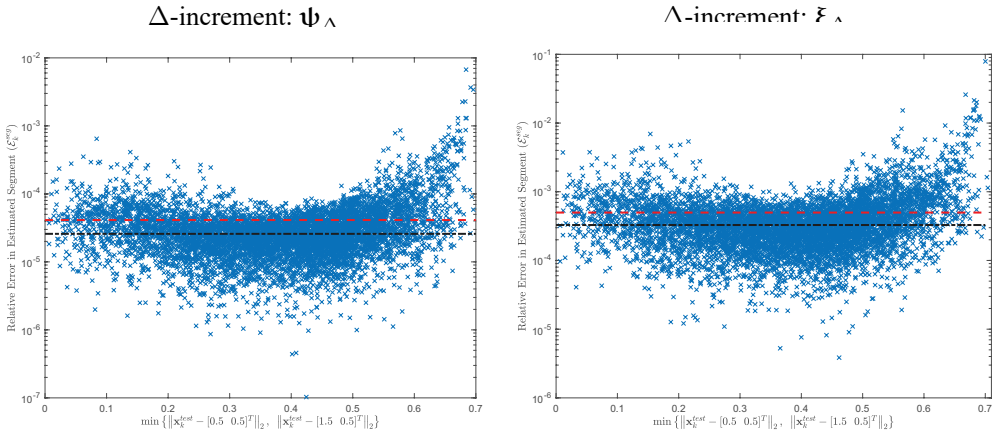


FIG. 14: Box plots of the segment relative error $\mathcal{E}_i^{\text{seg}}$ (19) in the double-gyre example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) The NNs for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ with fixed $\Delta = T/200 = 0.02$ and four different network structures (see Table 6). (b) The NNs for $\boldsymbol{\psi}_\Delta$ with $N_{\text{lay}} = 3$, $N_{\text{neu}} = 45$, and five different values of Δ ($T/25 = 0.16$, $T/50 = 0.08$, $T/100 = 0.04$, $T/200 = 0.02$, and $T/400 = 0.01$). (c) The NNs for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ with fixed $\Delta = T/200 = 0.02$ and four different network structures (see Table 7). (d) The NNs for $\boldsymbol{\xi}_\Delta$ with $N_{\text{lay}} = 3$, $N_{\text{neu}} = 45$ and five different values of Δ ($T/25 = 0.16$, $T/50 = 0.08$, $T/100 = 0.04$, $T/200 = 0.02$, and $T/400 = 0.01$).

point of the kth segment in the test set (17), $\mathbf{x}_k^{\text{test}}$, and $[0.5 \;\; 0.5]^T$, and 2) $\left\|\mathbf{x}_k^{\text{test}} - [1.5 \;\; 0.5]^T\right\|_2$, that is, the distance between $\mathbf{x}_k^{\text{test}}$ and $[1.5 \;\; 0.5]^T$, where $[0.5 \;\; 0.5]^T$ and $[1.5 \;\; 0.5]^T$ are roughly the gyre centers at $t = 0$ and $t = 0.5$ [see Figs. 13(a) and 13(c)]. The mean and median of $\mathcal{E}_k^{\text{seg}}$ are also displayed in each plot for reference. We observe that for both NNs, the errors loosely form a U shape, indicating that large errors occur either close to the center or the edges of the gyres; in particular, the largest errors correspond to segments furthest away from the gyres, at the boundary of the domain.



(a) (b)

FIG. 15: The segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) against $\min\{\|\mathbf{x}_k^{\text{test}} - [0.5\ \ 0.5]^T\|_2,\ \|\mathbf{x}_k^{\text{test}} - [1.5\ \ 0.5]^T\|_2\}$ in the double-gyre example, where $\mathbf{x}_k^{\text{test}}$ is the starting point of the kth segment in the test set (17), and $[0.5\ \ 0.5]^T$, $[1.5\ \ 0.5]^T$ are roughly the gyre centers at $t = 0$ and $t = 0.5$ [see Figs. 13(a) and 13(c)]. A semi-log scale is used. The horizontal dashed line marks the mean $\mathcal{E}_k^{\text{seg}}$. The horizontal dash-dotted line marks the median of $\mathcal{E}_k^{\text{seg}}$. (a) The NN with $\Delta = T/200 = 0.02$, $N_{\text{lay}} = 3$, and $N_{\text{neu}} = 30$ for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$. (b) The NN with $\Delta = T/200 = 0.02$, $N_{\text{lay}} = 3$, and $N_{\text{neu}} = 45$ for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$.

As in the previous two examples, we follow Steps 7–9 to examine how well Algorithm 1 approximates complete fluid particle trajectories. Recall that for five values of Δ, we have built five NNs with $N_{\text{lay}} = 3$, $N_{\text{neu}} = 30$ for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$, and five NNs with $N_{\text{lay}} = 3$, $N_{\text{neu}} = 45$ for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ [see Tables 6, 7, and Figs. 14(b) and 14(d) for their performance]. We first simulate N_{traj} trajectories with end time T in Step 7, and then repeat Steps 8 and 9 for every version of Algorithm 1 equipped with one of the 10 trained NNs, that is, Algorithm 1 ($\boldsymbol{\psi}_\Delta$, Δ, 3, 30, T) and Algorithm 1 ($\boldsymbol{\xi}_\Delta$, Δ, 3, 45, T) for the five values of Δ considered in Figs. 14(b) and 14(d). The statistics of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with every version of Algorithm 1 are depicted in Figs. 16(a) and 16(c). Similar to what has been observed for the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) from Figs. 14(b) and 14(d), for every Δ, the mean/median $\mathcal{E}_i^{\text{traj}}$ associated with Algorithm 1 ($\boldsymbol{\xi}_\Delta$, Δ, 3, 45, T) is about one order of magnitude higher than the mean/median $\mathcal{E}_i^{\text{traj}}$ associated with Algorithm 1 ($\boldsymbol{\psi}_\Delta$, Δ, 3, 30, T), even though the NN model used in the former has about twice as many network parameters as the NN model used in the latter (see Tables 6 and 7).

In Fig. 17(a), we also show the fifty most-inaccurate trajectories (marked by circles) estimated using Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, T). The exact trajectories are shown in solid lines for comparison. All fifty pairs of trajectories are near the boundary of the domain D, which is consistent with our observation based on Fig. 15 that the NN models are less accurate away from the gyre centers. In Fig. 17(b), we zoom in on the five least-accurate estimated trajectories and the true trajectories corresponding to them. The errors $\mathcal{E}_i^{\text{traj}}$ (21) associated with them are also displayed.

Next, we extend the trajectory end time T' in Step 7 to as large as $2T$ and examine the performance of Algorithm 1 again. The end time used in Step 2 to acquire the data set (18)

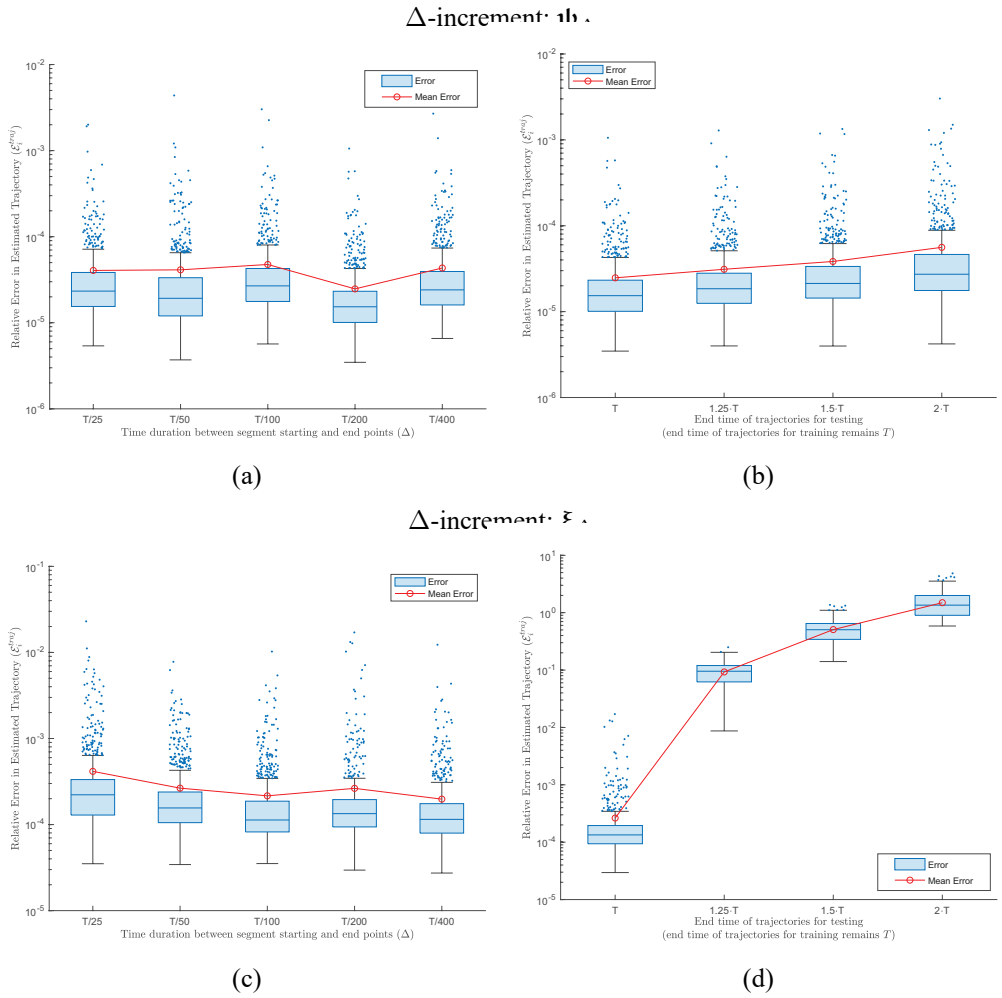
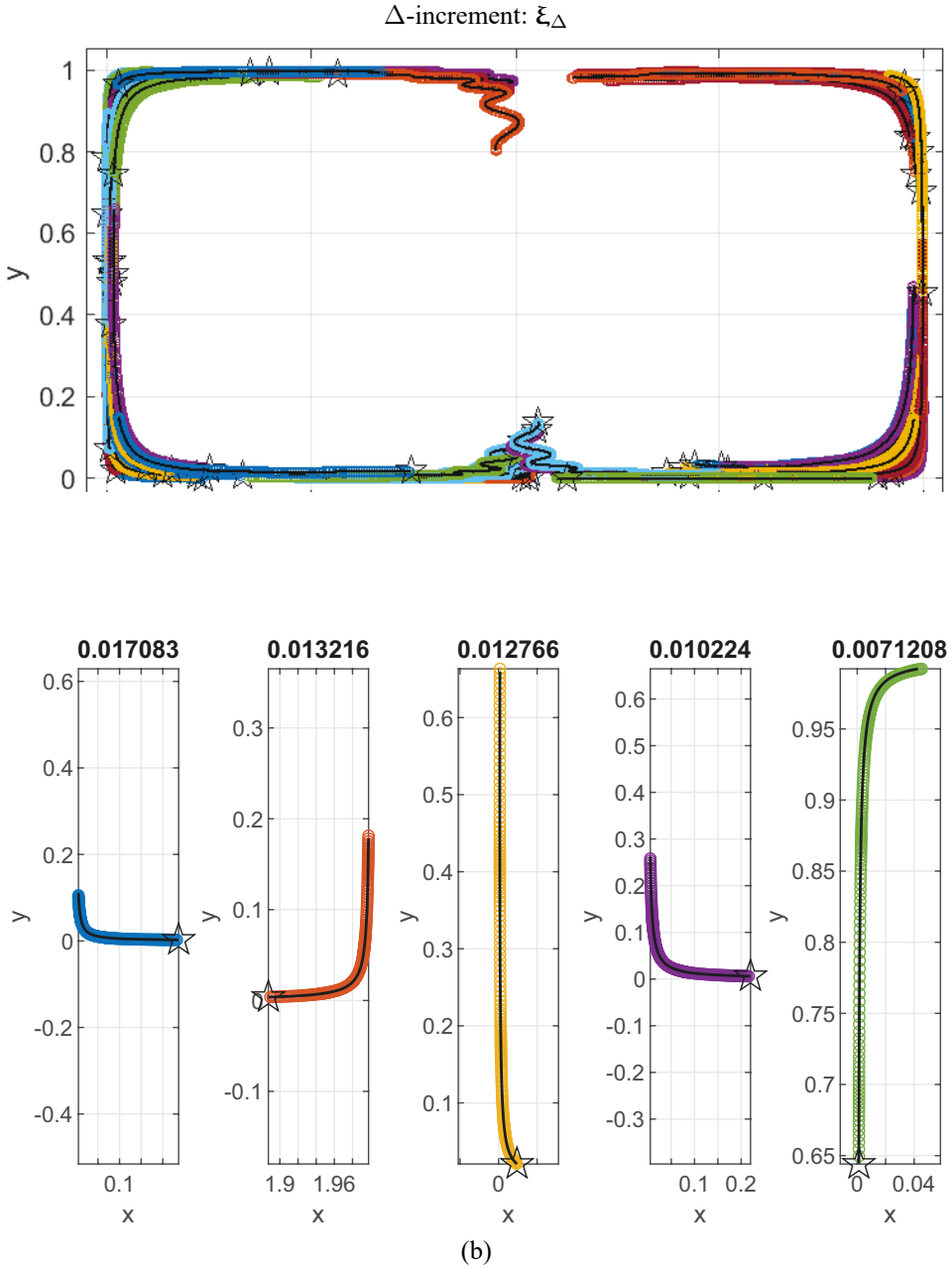


FIG. 16: Box plots of the trajectory relative error (21) associated with Algorithm 1 in the double-gyre example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) Algorithm 1 ($\boldsymbol{\psi}_\Delta$, Δ, 3, 30, T) where Δ equals $T/25 = 0.16$, $T/50 = 0.08$, $T/100 = 0.04$, $T/200 = 0.02$, or $T/400 = 0.01$. (b) Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30, T') where T' equals $T = 4$, $1.25T = 5$, $1.5T = 6$, or $2T = 8$. (c) Algorithm 1 ($\boldsymbol{\xi}_\Delta$, Δ, 3, 45, T) where Δ equals $T/25 = 0.16$, $T/50 = 0.08$, $T/100 = 0.04$, $T/200 = 0.02$, or $T/400 = 0.01$. (d) Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, T') where T' equals $T = 4$, $1.25T = 5$, $1.5T = 6$, or $2T = 8$.

for the NNs remains T and does not change. For each end time T' considered, we estimate the N_{traj} trajectories simulated in Step 7 using two versions of Algorithm 1: Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30, T') and Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, T'). The trajectory relative errors $\mathcal{E}_i^{\text{traj}}$ (21) associated with them are illustrated in Figs. 16(b) and 16(d), respectively. As expected, for both versions of Algorithm 1, $\mathcal{E}_i^{\text{traj}}$ grows as the end time T' increases beyond T. However, comparing the two figures, we notice that the growth of $\mathcal{E}_i^{\text{traj}}$ is considerably more rapid in the case of Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, T') than in the case of Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30,



(b)

FIG. 17: The fifty most-inaccurate trajectories estimated by Algorithm 1 (ξ_Δ, $T/200$, 3, 45, T) and the corresponding true trajectories in the double-gyre example. The circles mark the estimated trajectories, and the solid lines represent the true trajectories. The starting point of each trajectory is marked with a star. (a) All 50 pairs of trajectories. (b) The five most-inaccurate estimated trajectories and the corresponding true trajectories. The trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) in each estimated trajectory is displayed at the top of the respective panel.

T'). In particular, the mean/median/maximum $\mathcal{E}_i^{\text{traj}}$ associated with Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, $1.25T$) is already of order 10^{-1}, suggesting that Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, T') should not be applied to trace fluid particles beyond the end time $T' = 1.25T$. In stark contrast, as observed in Fig. 16(b), for T' as large as $2T$, the mean/median $\mathcal{E}_i^{\text{traj}}$ associated with Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30, T') is below 10^{-4}, and the maximum $\mathcal{E}_i^{\text{traj}}$ associated with it is below 10^{-2}, demonstrating the overwhelming superiority of the NN models for $\boldsymbol{\psi}_\Delta$ over the NN models for $\boldsymbol{\xi}_\Delta$ at extrapolating trajectory data. (See Appendix C for additional graphics.)

3.4 Unsteady ABC Flow

The unsteady ABC flow (Haller, 2005) resembles the steady ABC flow but incorporates time-dependent oscillations in the velocity field.

The velocity field $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ where $\boldsymbol{u} = [u \;\; v \;\; w]^T$ and $\boldsymbol{x} = [x \;\; y \;\; z]^T$ is as follows:

$$u(\boldsymbol{x}, \gamma(t)) = A(t) \sin z + C \cos y, \tag{34}$$

$$v(\boldsymbol{x}, \gamma(t)) = B \sin x + A(t) \cos z, \tag{35}$$

$$w(\boldsymbol{x}, \gamma(t)) = C \sin y + B \cos x, \tag{36}$$

where $A(t) = A_0 + (1 - e^{-qt}) \sin \omega t$ characterizes the growth and saturation of an unstable mode, and $\gamma(t)$ coincides with $A(t)$. The parameter values are $A_0 = \sqrt{3}$, $q = 0.1$, $\omega = 2\pi$, $B = \sqrt{2}$, and $C = 1$ as in Haller (2005). Since this flow is also unsteady, we proceed as in the previous example. One major difference is that the flow is 3D in this example. In Steps 1 and 7, we sample the particles within the cube $D = [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$ and let the edge length 2π be the characteristic length. The largest amplitude in all three components of the velocity, $A_0 + B$, is chosen to be the characteristic speed. Consequently, as in Section 3.2 for the steady ABC flow, we employ $T = 2$ as the characteristic time, which is approximately $2\pi/(A_0 + B)$.

As in Section 3.3, we first follow Steps 1–6 to examine the performance of NN models for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ and the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$. The time-dependent term γ in the fluid velocity vector field (34)–(36) is again not a polynomial. For the starting time point t_k of each trajectory segment in (18) selected in Step 3, we again use the second-degree Taylor polynomial of γ around t_k to approximate γ locally on that segment, between time points t_k and $t_k + \Delta$. Since the flow is 3D, according to Table 1, $\boldsymbol{\psi}_\Delta$ is from $\mathbb{R}^6$ to $\mathbb{R}^3$, and $\boldsymbol{\xi}_\Delta$ is from $\mathbb{R}^4$ to $\mathbb{R}^3$. Various network structures and values of Δ are again considered. The statistics of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) are summarized in Tables 8, 9, and Fig. 18. As seen in Figs. 18(a) and 18(c), the network structure plays an important role in the accuracy of NN models for both $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$. When the total number of network parameters, N_{param}, remains more or less the same, the NN for $\boldsymbol{\psi}_\Delta$ with two hidden layers ($N_{\text{lay}} = 2$) is the most accurate, while the NN for $\boldsymbol{\xi}_\Delta$ with one hidden layer ($N_{\text{lay}} = 1$) is the most accurate. Figures 18(b) and 18(d) suggest that the accuracy of NN models is not very sensitive to the value of Δ. For every Δ considered, the mean/median $\mathcal{E}_k^{\text{seg}}$ is in the order of 10^{-4} for the NN models of both $\boldsymbol{\psi}_\Delta$ and $\boldsymbol{\xi}_\Delta$.

As in the previous three examples, we follow Steps 7–9 to examine the performance of Algorithm 1 at approximating fluid particle trajectories. For five values of Δ, we report the statistics of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with Algorithm 1 ($\boldsymbol{\psi}_\Delta$, Δ, 2, 55, T) and Algorithm 1 ($\boldsymbol{\xi}_\Delta$, Δ, 1, 400, T) in Figs. 19(a) and 19(c). The median/mean of $\mathcal{E}_i^{\text{traj}}$ is in the order of 10^{-4} across the board.

TABLE 8: The minimum, mean, and maximum segment relative errors $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ in the unsteady ABC example. $\Delta = T/200 = 0.01$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu} and N_{param})

N_{lay}	N_{neu}	N_{param}	Min. $\mathcal{E}_k^{\text{seg}}$	Mean $\mathcal{E}_k^{\text{seg}}$	Max. $\mathcal{E}_k^{\text{seg}}$
1	400	4003	$1.62 \cdot 10^{-5}$	$5.77 \cdot 10^{-4}$	$2.19 \cdot 10^{-2}$
2	55	3633	$2.58 \cdot 10^{-6}$	$1.07 \cdot 10^{-4}$	$7.11 \cdot 10^{-3}$
3	40	3683	$6.04 \cdot 10^{-6}$	$2.36 \cdot 10^{-4}$	$1.27 \cdot 10^{-2}$
5	30	4023	$6.83 \cdot 10^{-6}$	$4.78 \cdot 10^{-4}$	$1.91 \cdot 10^{-2}$

TABLE 9: The minimum, mean, and maximum segment relative errors $\mathcal{E}_k^{\text{seg}}$ (19) for four NN models of the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ in the unsteady ABC example. $\Delta = T/200 = 0.01$ is fixed. (See Table 3 for the meaning of N_{lay}, N_{neu} and N_{param})

N_{lay}	N_{neu}	N_{param}	Min. $\mathcal{E}_k^{\text{seg}}$	Mean $\mathcal{E}_k^{\text{seg}}$	Max. $\mathcal{E}_k^{\text{seg}}$
1	400	3604	$2.06 \cdot 10^{-6}$	$1.04 \cdot 10^{-4}$	$3.29 \cdot 10^{-2}$
2	55	3579	$1.28 \cdot 10^{-5}$	$5.29 \cdot 10^{-4}$	$1.93 \cdot 10^{-2}$
3	40	3644	$5.01 \cdot 10^{-6}$	$4.28 \cdot 10^{-4}$	$1.40 \cdot 10^{-2}$
5	30	3994	$2.22 \cdot 10^{-6}$	$8.74 \cdot 10^{-4}$	$2.52 \cdot 10^{-2}$

Finally, we apply Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/400$, 2, 55, T') and Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/400$, 1, 400, T') to estimate trajectories with end time T' between T and $2T$. Recall that T is the end time of the trajectories from which the training data (15) are collected. For each T' considered, the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) is reported in Figs. 19(b) and 19(d). As expected, whether an NN model for $\boldsymbol{\psi}_\Delta$ or $\boldsymbol{\xi}_\Delta$ is employed in Algorithm 1, $\mathcal{E}_i^{\text{traj}}$ increases with T'. It is again evident that the NN models for $\boldsymbol{\xi}_\Delta$ are worse at extrapolating than the NN models for $\boldsymbol{\psi}_\Delta$. For example, the median $\mathcal{E}_i^{\text{traj}}$ is about 10^{-2} for Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/400$, 1, 400, $1.5T$) and less than 10^{-3} for Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/400$, 1, 400, $1.5T$).

In this example, the NN models for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ are as accurate as the NN models for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ at interpolating, when the time domains of the training data and test data coincide [see Tables 8, 9, Fig. 18, and Figs. 19(a) and 19(c)]. However, the former are less accurate at extrapolating, when the test data are collected outside of the time domain of the training data [see Figs. 19(b) and 19(d)]. We emphasize again that training an NN for $\boldsymbol{\xi}_\Delta$ does not require any knowledge of the time-dependent term γ in the fluid velocity, which is a huge advantage in scenarios where only trajectory data are available.

4. DISCUSSION

The position of a fluid particle is governed by an autonomous or a non-autonomous dynamical system, depending on whether the the flow is steady or unsteady. This system can be simulated using a numerical ODE solver, given that the fluid velocity field is known or can be resolved. Our main contribution is demonstrating that given abundant trajectory data, we can learn the solution to this system and thus track the fluid particle without knowledge of the fluid velocity.

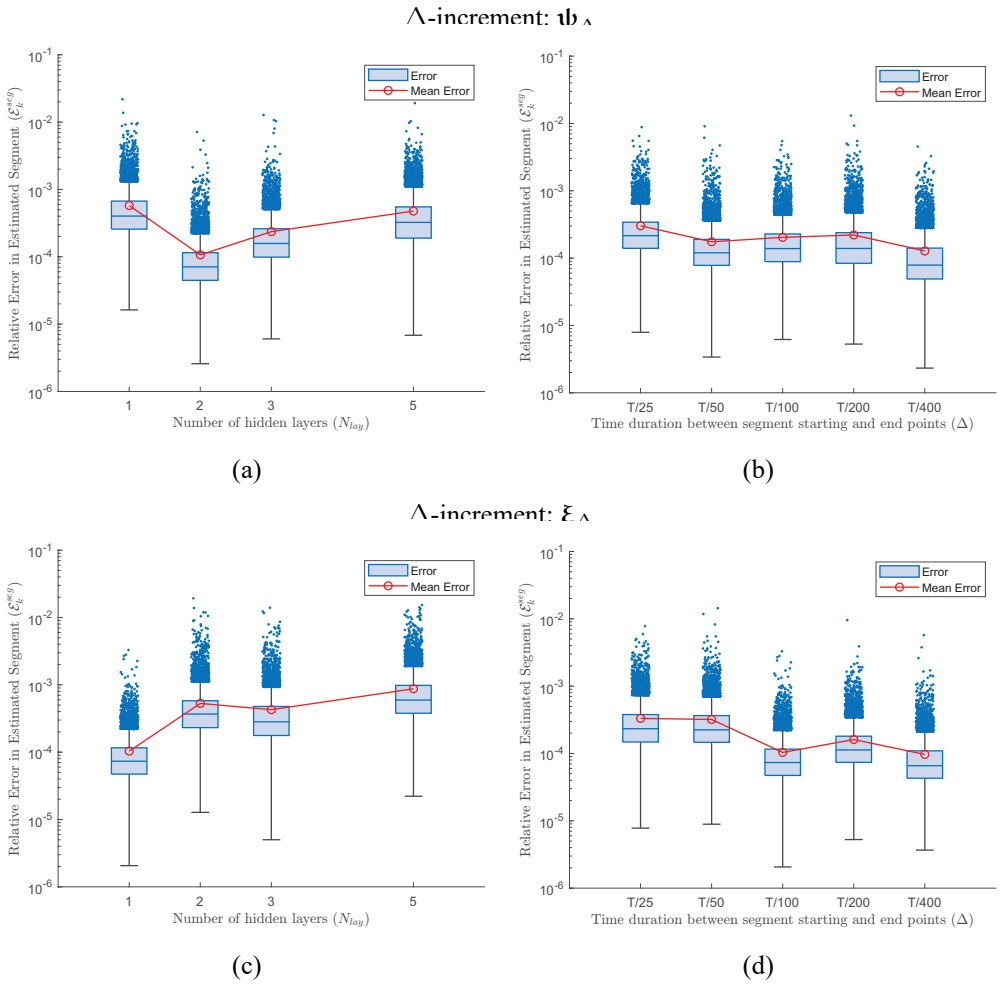


FIG. 18: Box plots of the segment relative error $\mathcal{E}_k^{\text{seg}}$ (19) in the unsteady ABC example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) The NNs for the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ with fixed $\Delta = T/100 = 0.02$ and four different network structures (see Table 8). (b) The NNs for $\boldsymbol{\psi}_\Delta$ with $N_{\text{lay}} = 2$, $N_{\text{neu}} = 55$ and five different values of Δ ($T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, and $T/200 = 0.01$, and $T/400 = 0.005$). (c) The NNs for the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ with fixed $\Delta = T/100 = 0.02$ and four different network structures (see Table 9). (d) The NNs for $\boldsymbol{\xi}_\Delta$ with $N_{\text{lay}} = 1$, $N_{\text{neu}} = 400$ and five different values of Δ ($T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, and $T/200 = 0.01$, and $T/400 = 0.005$).

Our work is based on and inspired by Qin et al. (2019, 2021), where DNNs were used to approximate the flow map of an autonomous system (Qin et al., 2019) or a non-autonomous system (Qin et al., 2021), which, given the solution to the system at one time point, produces the solution at a later time point. The flow map was built on an increment function that outputs the change in the solution between the two time points. One of our contributions is finding novel analytic expressions for the increment function for an autonomous system (Qin et al., 2019) and the one for a non-autonomous system (Qin et al., 2021), termed the steady Δ-increment and the

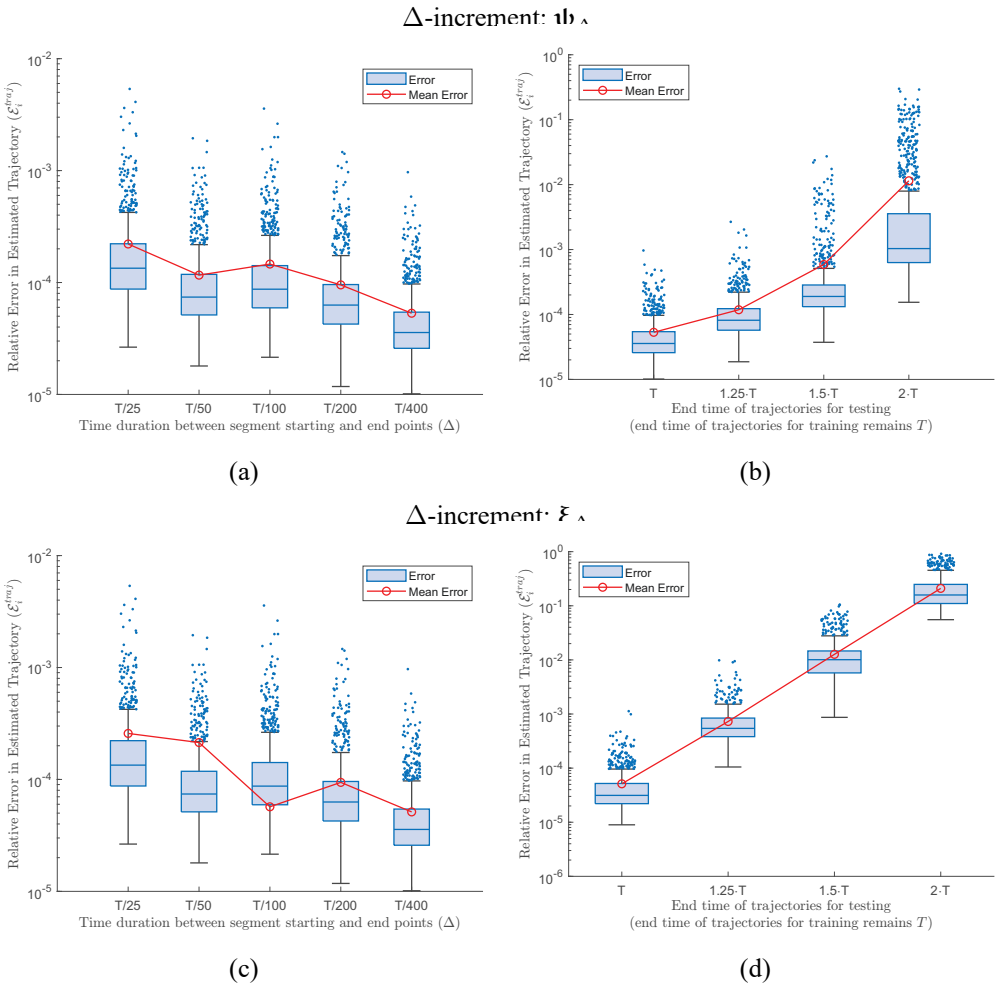


FIG. 19: Box plots of the trajectory relative error $\mathcal{E}_i^{\text{traj}}$ (21) associated with Algorithm 1 in the unsteady ABC flow example. A log scale is used on the vertical axis. (See the caption of Fig. 3 for interpretation of box plots.) (a) Algorithm 1 (ψ_Δ, Δ, 2, 55, T) where Δ equals $T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, $T/200 = 0.01$, or $T/400 = 0.005$. (b) Algorithm 1 (ψ_Δ, 0.005, 1, 400, T') where T' equals $T = 2$, $1.25T = 2.5$, $1.5T = 3$, or $2T = 4$. (c) Algorithm 1 (ξ_Δ, Δ, 2, 55, T) where Δ equals $T/25 = 0.08$, $T/50 = 0.04$, $T/100 = 0.02$, $T/200 = 0.01$, or $T/400 = 0.005$. (d) Algorithm 1 (ξ_Δ, 0.005, 1, 400, T') where T' equals $T = 2$, $1.25T = 2.5$, $1.5T = 3$, or $2T = 4$.

γ-explicit unsteady Δ-increment in this work, and thus providing new insights into how they relate to the forcing terms in these dynamical systems.

A disadvantage of the γ-explicit unsteady Δ-increment is that to learn it, we need to know the time-dependent term γ in the velocity field in addition to trajectory data, which can be unrealistic in real-world applications where an analytic expression of the velocity field is truly unknown and only trajectory data are available. To remedy this, we propose a new increment function for a non-autonomous system that can be learned from time-stamped trajectory data alone, termed the γ-implicit unsteady Δ-increment. This is another contribution of our work.

Numerical results show that the DNNs built for the γ-explicit unsteady Δ-increment are more accurate and also better at extrapolating trajectory data, compared to the DNN models of the proposed γ-implicit unsteady Δ-increment. This is not surprising since the former are also informed by additional knowledge of the flow (specifically, how the fluid velocity changes with time) besides the trajectory data.

Furthermore, we look into how the physical features of a fluid flow affect the accuracy of a DNN model for the increment function. For example, we observe that near the boundary of a vortex, the model is the least accurate. Such information could lead to more clever sampling of the trajectory data as well as adaptive refinement of the model that improves its performance in certain regions of the flow.

One limitation of this work is that the fluid velocity is known analytically in all examples considered. This allows for computationally cheap simulation of trajectory data but fails to account for the noises that will inevitably arise from resolving the fluid velocity field by a numerical method for the PDEs governing the fluid dynamics, such as the Navier-Stokes equations, or by a flow measurement technique such as the PIV. Our future directions include using more realistic trajectory data generated by these means.

ACKNOWLEDGMENTS

Wei and Rostami were supported, in part, by the National Science Foundation (NSF) under Grants DMS-1818833 and DMS-2146191, 2408964 (CAREER) awarded to Rostami. Green was supported by the Computational Mathematics program of the Air Force Office of Scientific Research under Grant FA9550-07-1-0139. Shen was supported by the NSF under Grant DMS-2208385.

REFERENCES

Brunton, S.L., Noack, B.R., and Koumoutsakos, P., Machine Learning for Fluid Mechanics, *Ann. Rev. Fluid Mech.*, vol. **52**, no. 1, pp. 477–508, 2020.

Chen, Z., Churchill, V., Wu, K., and Xiu, D., Deep Neural Network Modeling of Unknown Partial Differential Equations in Nodal Space, *J. Comput. Phys.*, vol. **449**, p. 110782, 2022.

Chen, Z. and Xiu, D., On Generalized Residual Network for Deep Learning of Unknown Dynamical Systems, *J. Comput. Phys.*, vol. **438**, p. 110362, 2021.

Elman, H., Silvester, D., and Wathen, A., *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, 2nd ed., Oxford, UK: Oxford University Press, 2014.

Fu, X., Chang, L.B., and Xiu, D., Learning Reduced Systems via Deep Neural Networks with Memory, *J. Mach. Learn. Model. Comput.*, vol. **1**, no. 2, pp. 97–118, 2020.

Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, Cambridge, MA: MIT Press, 2016.

Haller, G., An Objective Definition of a Vortex, *J. Fluid Mech.*, vol. **525**, pp. 1–26, 2005.

Haller, G., Lagrangian Coherent Structures, *Ann. Rev. Fluid Mech.*, vol. **47**, no. 1, pp. 137–162, 2015.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, pp. 770–778, 2016.

Hunt, J.C.R., Wray, A.A., and Moin, P., Eddies, Streams, and Convergence Zones in Turbulent Flows, in *Proc. of the Summer Program*, Center for Turbulence Research, pp. 193–208, 1988.

Jain, A., Mao, J., and Mohiuddin, K., Artificial Neural Networks: A Tutorial, *Computer*, vol. **29**, no. 3, pp. 31–44, 1996.

Lin, G., Moya, C., and Zhang, Z., On Learning the Dynamical Response of Nonlinear Control Systems with Deep Operator Networks, arXiv:2206.06536[math.DS], 2023.

Long, Z., Lu, Y., Ma, X., and Dong, B., PDE-Net: Learning PDEs from Data, in *Proc. of the 35th Intl. Conf. on Machine Learning*, Vol. 80 of *Proc. of Machine Learning Research*, Stockholm, Sweden, pp. 3208–3216, 2018.

Long, Z., Lu, Y., and Dong, B., PDE-Net 2.0: Learning PDEs from Data with a Numeric-Symbolic Hybrid Deep Network, *J. Comput. Phys.*, vol. **399**, p. 108925, 2019.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G.E., Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators, *Nat. Mach. Intell.*, vol. **3**, pp. 218–229, 2021.

Marquardt, D.W., An Algorithm for Least-Squares Estimation of Nonlinear Parameters, *J. Soc. Indust. Appl. Math.*, vol. **11**, no. 2, pp. 431–441, 1963.

Proctor, J.L., Brunton, S.L., and Kutz, J.N., Dynamic Mode Decomposition with Control, *SIAM J. Appl. Dyn. Syst.*, vol. **15**, no. 1, pp. 142–161, 2016.

Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, *J. Comput. Phys.*, vol. **395**, pp. 620–635, 2019.

Qin, T., Chen, Z., Jakeman, J.D., and Xiu, D., Data-Driven Learning of Nonautonomous Systems, *SIAM J. Sci. Comput.*, vol. **43**, no. 3, pp. A1607–A1624, 2021.

Raffel, M., Willert, C.E., Scarano, F., Kähler, C.J., Wereley, S.T., and Kompenhans, J., *Particle Image Velocimetry: A Practical Guide*, 3rd ed., Cham: Springer, 2018.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Multistep Neural Networks for Data-Driven Discovery of Nonlinear Dynamical Systems, arXiv:1801.01236[math.DS], 2018.

Rockwood, M.P., Loiselle, T., and Green, M.A., Practical Concerns of Implementing a Finite-Time Lyapunov Exponent Analysis with Under-Resolved Data, *Exp. Fluids*, vol. **60**, no. 4, p. 74, 2019.

Schanz, D., Gesemann, S., and Schröder, A., Shake-the-Box: Lagrangian Particle Tracking at High Particle Image Densities, *Exp. Fluids*, vol. **57**, no. 5, p. 70, 2016.

Shadden, S.C., Lekien, F., and Marsden, J.E., Definition and Properties of Lagrangian Coherent Structures from Finite-Time Lyapunov Exponents in Two-Dimensional Aperiodic Flows, *Physica D: Nonlinear Phenomena*, vol. **212**, no. 3, pp. 271–304, 2005.

Su, W.H., Chou, C.S., and Xiu, D., Deep Learning of Biological Models from Data: Applications to ODE Models, *Bull. Math. Biol.*, vol. **83**, no. 3, p. 19, 2021.

Tang, B., Orthogonal Array-Based Latin Hypercubes, *J. Am. Stat. Assoc.*, vol. **88**, no. 424, pp. 1392–1397, 1993.

van Sebille, E., Griffies, S.M., Abernathey, R., Adams, T.P., Berloff, P., Biastoch, A., Blanke, B., Chassignet, E.P., Cheng, Y., Cotter, C.J., Deleersnijder, E., Döös, K., Drake, H.F., Drijfhout, S., Gary, S.F., Heemink, A.W., Kjellsson, J., Koszalka, I.M., Lange, M., Lique, C., MacGilchrist, G.A., Marsh, R., Mayorga Adame, C.G., McAdam, R., Nencioli, F., Paris, C.B., Piggott, M.D., Polton, J.A., Rühs, S., Shah, S.H.A.M., Thomas, M.D., Wang, J., Wolfram, P.J., Zanna, L., and Zika, J.D., Lagrangian Ocean Analysis: Fundamentals and Practices, *Ocean Model.*, vol. **121**, pp. 49–75, 2018.

Wu, K., Qin, T., and Xiu, D., Structure-Preserving Method for Reconstructing Unknown Hamiltonian Systems from Trajectory Data, *SIAM J. Sci. Comput.*, vol. **42**, no. 6, pp. A3704–A3729, 2020.

Wu, K. and Xiu, D., Data-Driven Deep Learning of Partial Differential Equations in Modal Space, *J. Comput. Phys.*, vol. **408**, p. 109307, 2020.

Zhuang, Q., Lorenzi, J.M., Bungartz, H.J., and Hartmann, D., Model Order Reduction Based on Runge–Kutta Neural Networks, *Data-Centric Eng.*, vol. **2**, p. e13, 2021.

APPENDIX A. PROOF OF THE THEOREMS

In this section and the next, we assume that the dimension of the flow, d_{flo}, is 3. In order to prove Theorem 1 for the autonomous system Eq. (1), we first prove the following lemma.

Lemma 1. *Let $\{\boldsymbol{u}_k\}_{k=1}^{\infty}$ be a sequence of functions from $\mathbb{R}^3$ to $\mathbb{R}^3$ defined as follows:*

1. *$\boldsymbol{u}_1 = \boldsymbol{u}$, where $\boldsymbol{u}$ is the fluid velocity on the right-hand side of Eq.* (1)*;*
2. *for any integer $k > 1$, $\boldsymbol{u}_k = J_{k-1}\boldsymbol{u}$, where J_{k-1} is the 3×3 Jacobian matrix of $\boldsymbol{u}_{k-1}$.*

Then for any $\mathbf{x}$ satisfying Eq. (1) *and any integer $k \geq 1$,*

$$\frac{\mathrm{d}^k \mathbf{x}}{\mathrm{d}t^k} = \boldsymbol{u}_k(\mathbf{x}(t)). \tag{A.1}$$

The definition of J_k is as follows:

$$J_k = \begin{bmatrix} \dfrac{\partial u_k}{\partial x} & \dfrac{\partial u_k}{\partial y} & \dfrac{\partial u_k}{\partial z} \\ \dfrac{\partial v_k}{\partial x} & \dfrac{\partial v_k}{\partial y} & \dfrac{\partial v_k}{\partial z} \\ \dfrac{\partial w_k}{\partial x} & \dfrac{\partial w_k}{\partial y} & \dfrac{\partial w_k}{\partial z} \end{bmatrix}, \tag{A.2}$$

where $\boldsymbol{u}_k = [u_k \;\; v_k \;\; w_k]^T$ and $\boldsymbol{x} = [x \;\; y \;\; z]^T$.

Proof. — For $k = 1$, Eq. (A.1) follows immediately from the definition of $\boldsymbol{u}_1$ and Eq. (1).

— Assume that for $k = m \geq 2$, Eq. (A.1) holds, that is,

$$\frac{\mathrm{d}^m \mathbf{x}}{\mathrm{d}t^m} = \boldsymbol{u}_m(\mathbf{x}(t)). \tag{A.3}$$

By the definition of $\boldsymbol{u}_k$, the definition of J_k, Eq. (1), Eq. (A.3), and the chain rule,

$$\frac{\mathrm{d}^{m+1}\mathbf{x}}{\mathrm{d}t^{m+1}} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}^m \mathbf{x}}{\mathrm{d}t^m}\right) = \frac{\mathrm{d}\boldsymbol{u}_m}{\mathrm{d}t} = J_m \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = J_m \boldsymbol{u} = \boldsymbol{u}_{m+1},$$

that is, Eq. (A.1) holds for $k = m + 1$ as well.

— By the principle of mathematical induction, Lemma 1 holds.

□

We are now ready to prove Theorem 1.

Proof. Let t_0 be an arbitrary, fixed time point in $[0, T - \Delta]$. Using the Taylor expansion of $\mathbf{x}(t)$ around t_0, we get

$$\mathbf{x}(t_0 + \Delta) - \mathbf{x}(t_0) = \Delta \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\bigg|_{t_0} + \frac{\Delta^2}{2}\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2}\bigg|_{t_0} + \frac{\Delta^3}{6}\frac{\mathrm{d}^3\mathbf{x}}{\mathrm{d}t^3}\bigg|_{t_0} + \cdots. \tag{A.4}$$

By Lemma 1, Eq. (A.4) can be rewritten as

$$\mathbf{x}(t_0 + \Delta) - \mathbf{x}(t_0) = \sum_{k=1}^{\infty} \frac{\Delta^k}{k!} \boldsymbol{u}_k(\mathbf{x}(t_0)) \equiv \boldsymbol{\phi}_{\Delta}(\mathbf{x}(t_0)) \tag{A.5}$$

for any $\mathbf{x}$ satisfying Eq. (1). Since t_0 is arbitrary, Eq. (2) and Theorem 1 follow immediately. □

In the case of the non-autonomous system Eq. (4), depending on whether we rewrite it into the autonomous system Eq. (6) or the autonomous system Eq. (10), we can prove Theorem 2 or Theorem 3 in a similar fashion.

APPENDIX B. EXPRESSIONS OF THE Δ-INCREMENTS

We note that the proof of Theorem 1 is constructive. From Eq. (A.5), we obtain the following expression of the steady Δ-increment $\boldsymbol{\phi}_\Delta$ in terms of $\{\boldsymbol{u}_k\}_{k=1}^{\infty}$:

$$\boldsymbol{\phi}_\Delta = \sum_{k=1}^{\infty} \frac{\Delta^k}{k!} \boldsymbol{u}_k. \tag{B.1}$$

Similarly, we can write down an expression of the γ-explicit unsteady Δ-increment $\boldsymbol{\psi}_\Delta$ in terms of $\{\widetilde{\boldsymbol{u}}_k\}_{k=1}^{\infty}$ defined as follows:

1. $\widetilde{\boldsymbol{u}}_1 = \boldsymbol{u}$, where $\boldsymbol{u}$ the fluid velocity on the right-hand side of Eq. (4);
2. for any integer $k > 1$, $\widetilde{\boldsymbol{u}}_k = \widetilde{J}_{k-1} \begin{bmatrix} \boldsymbol{u} \\ \begin{bmatrix} \gamma^{(1)} & \gamma^{(2)} & \cdots & \gamma^{(m)} & 0 \end{bmatrix}^T \end{bmatrix}$, where

$$\widetilde{J}_k = \begin{bmatrix} \dfrac{\partial \widetilde{u}_k}{\partial x} & \dfrac{\partial \widetilde{u}_k}{\partial y} & \dfrac{\partial \widetilde{u}_k}{\partial z} & \dfrac{\partial \widetilde{u}_k}{\partial \gamma} & \dfrac{\partial \widetilde{u}_k}{\partial \gamma^{(1)}} & \cdots & \dfrac{\partial \widetilde{u}_k}{\partial \gamma^{(m)}} \\ \dfrac{\partial \widetilde{v}_k}{\partial x} & \dfrac{\partial \widetilde{v}_k}{\partial y} & \dfrac{\partial \widetilde{v}_k}{\partial z} & \dfrac{\partial \widetilde{v}_k}{\partial \gamma} & \dfrac{\partial \widetilde{v}_k}{\partial \gamma^{(1)}} & \cdots & \dfrac{\partial \widetilde{v}_k}{\partial \gamma^{(m)}} \\ \dfrac{\partial \widetilde{w}_k}{\partial x} & \dfrac{\partial \widetilde{w}_k}{\partial y} & \dfrac{\partial \widetilde{w}_k}{\partial z} & \dfrac{\partial \widetilde{w}_k}{\partial \gamma} & \dfrac{\partial \widetilde{w}_k}{\partial \gamma^{(1)}} & \cdots & \dfrac{\partial \widetilde{w}_k}{\partial \gamma^{(m)}} \end{bmatrix}, \tag{B.2}$$

and $\widetilde{\boldsymbol{u}}_k = [\widetilde{u}_k \;\; \widetilde{v}_k \;\; \widetilde{w}_k]^T$.

This expression is

$$\boldsymbol{\psi}_\Delta = \sum_{k=1}^{\infty} \frac{\Delta^k}{k!} \widetilde{\boldsymbol{u}}_k. \tag{B.3}$$

We can also write down an expression of the γ-implicit unsteady Δ-increment $\boldsymbol{\xi}_\Delta$ in terms of $\{\widehat{\boldsymbol{u}}_k\}_{k=1}^{\infty}$ defined as follows:

1. $\widehat{\boldsymbol{u}}_1 = \boldsymbol{u}$, where $\boldsymbol{u}$ the fluid velocity on the right-hand side of Eq. (4);
2. for any integer $k > 1$, $\widehat{\boldsymbol{u}}_k = \widehat{J}_{k-1} \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}$, where

$$\widehat{J}_k = \begin{bmatrix} \dfrac{\partial \widehat{u}_k}{\partial x} & \dfrac{\partial \widehat{u}_k}{\partial y} & \dfrac{\partial \widehat{u}_k}{\partial z} & \dfrac{\partial \widehat{u}_k}{\partial t} \\ \dfrac{\partial \widehat{v}_k}{\partial x} & \dfrac{\partial \widehat{v}_k}{\partial y} & \dfrac{\partial \widehat{v}_k}{\partial z} & \dfrac{\partial \widehat{v}_k}{\partial t} \\ \dfrac{\partial \widehat{w}_k}{\partial x} & \dfrac{\partial \widehat{w}_k}{\partial y} & \dfrac{\partial \widehat{w}_k}{\partial z} & \dfrac{\partial \widehat{w}_k}{\partial t} \end{bmatrix}, \tag{B.4}$$

and $\widehat{\boldsymbol{u}}_k = [\widehat{u}_k \;\; \widehat{v}_k \;\; \widehat{w}_k]^T$.

This expression is

$$\boldsymbol{\xi}_{\Delta} = \sum_{k=1}^{\infty} \frac{\Delta^k}{k!} \widehat{\boldsymbol{u}}_k. \tag{B.5}$$

APPENDIX C. ADDITIONAL GRAPHICS OF THE PARTICLE TRAJECTORIES

We present additional figures that illustrate the growth of the error of Algorithm 1 with time. We consider one steady flow and one unsteady flow: the Hill's spherical vortex example (Section 3.1) and the double-gyre example (Section 3.3).

Recall that in Section 3.1, we apply Algorithm 1 ($\boldsymbol{\Phi}_{\Delta}$, $T/100$, 5, 20, T') to track 1000 particles from time 0 to T', where the end time T' varies between T and $2T$, and the NN model for $\boldsymbol{\Phi}_{\Delta}$ is trained on trajectory data collected on the time domain $[0, T]$ with $T = 8$ [see Fig. 5(b)]. For the end time $T' = 1.5T$, we plot the x, y, and z components of the least accurately estimated trajectory as well as their exact counterparts against time in Fig. C1. The relative error, Eq. (21), associated with this trajectory is about 1.04×10^{-1}. Even though T' is 1.5 times as large as T, the x and y components of the estimated and exact trajectories are almost indistinguishable (see the top and center panel of Fig. C1); the error in the z component of the estimated trajectory becomes noticeable as time approaches T'.

Recall that in Section 3.3, we apply both Algorithm 1 ($\boldsymbol{\psi}_{\Delta}$, $T/200$, 3, 30, T') and Algorithm 1 ($\boldsymbol{\xi}_{\Delta}$, $T/100$, 3, 45, T') to track 1000 particles from time 0 to T', where the end time T' varies between T and $2T$, and the NN models for $\boldsymbol{\psi}_{\Delta}$ and $\boldsymbol{\xi}_{\Delta}$ are trained on trajectory data collected on the time domain $[0, T]$ with $T = 4$ [see Figs. 16(b) and 16(d)]. Also recall that the

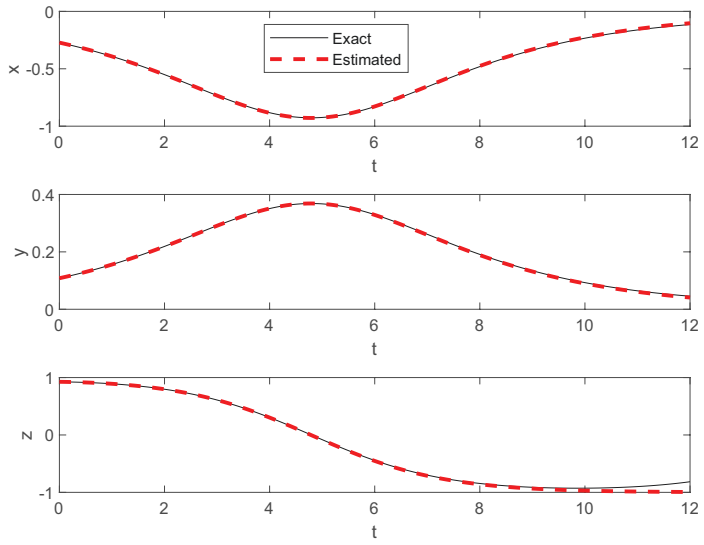


FIG. C1: The x, y, and z components of the least accurate trajectory estimated by Algorithm 1 ($\boldsymbol{\Phi}_{\Delta}$, $T/100$, 5, 20, $1.5T$) as functions of time (t) in the Hill's spherical vortex example (Section 3.1). The red dashed line represents the x, y, or z component of the estimated trajectory, and the solid black line represents the x, y, or z component of the exact trajectory. The trajectory end time is $1.5T$, whereas the training data are sampled from the time domain $[0, T]$ ($T = 8$).

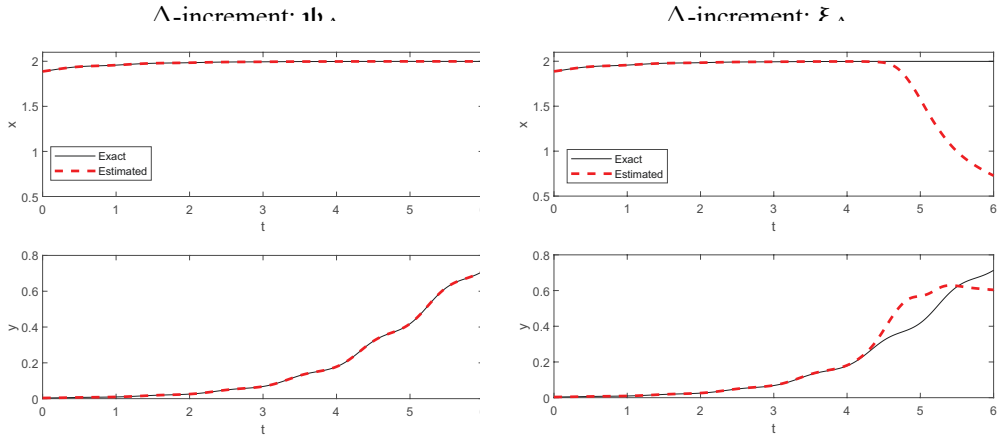


FIG. C2: The x and y components of the least-accurate trajectory estimated by Algorithm 1 as functions of time (t) in the double-gyre example (Section 3.3). The red dashed line represents the x or y component of the estimated trajectory, and the solid black line represents the x or y component of the exact trajectory. The trajectory end time is $1.5T$, whereas the training data are sampled from the time domain $[0, T]$ ($T = 4$). (a) Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30, $1.5T$). (b) Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, $1.5T$).

training of NN models for $\boldsymbol{\psi}_\Delta$ requires knowing the time-dependent term $\gamma(t)$ in the fluid velocity $\boldsymbol{u}(\boldsymbol{x}, \gamma(t))$ explicitly, whereas the training of NN models for $\boldsymbol{\xi}_\Delta$ does not. As in the previous example, we consider the end time $T' = 1.5T$. In Fig. C2(a), we plot the x and y components of the least-accurate trajectory estimated by Algorithm 1 ($\boldsymbol{\psi}_\Delta$, $T/200$, 3, 30, $1.5T$) as well as their exact counterparts against time. The relative error associated with this trajectory is about 1.3×10^{-3}. As displayed in Fig. C2(a), both the x and y components of the estimated trajectory are indistinguishable from the exact ones between time 0 and $1.5T$.

We also plot the x and y components of the least-accurate trajectory estimated by Algorithm 1 ($\boldsymbol{\xi}_\Delta$, $T/200$, 3, 45, $1.5T$) as well as their exact counterparts against time in Fig. C2(b). Both estimated components lose accuracy rapidly around time $1.1T$; and the relative error in the estimated trajectory is as large as 1.37. The comparison between Figs. C2(a) and C2(b) shows that although both the NN model for $\boldsymbol{\psi}_\Delta$ and the NN model for $\boldsymbol{\xi}_\Delta$ are able to make predictions beyond the time domain of the training data, the former, due to the additional knowledge of the time-dependent term $\gamma(t)$, remains accurate for a much longer period of time.