Journal of Flow Visualization & Image Processing, x(x): 1–18 (2017)

EXPLORATORY LAGRANGIAN-BASED PARTICLE TRACING USING DEEP LEARNING

Mengjiao Han,* Sudhanshu Sane, & Chris R. Johnson

University of Utah, SCI Institute, Salt Lake City, UT 84112, USA

*Address all correspondence to: Mengjiao Han, E-mail: mengjiao@sci.utah.edu

Time-varying vector fields produced by computational fluid dynamics simulations are often prohibitively large and pose challenges for accurate interactive analysis and exploration. To address these challenges, reduced Lagrangian representations have been increasingly researched as a means to improve scientific time-varying vector field exploration capabilities. This paper presents a novel deep neural network-based particle tracing method to explore time-varying vector fields represented by Lagrangian flow maps. In our workflow, in situ processing is first utilized to extract Lagrangian flow maps, and deep neural networks then use the extracted data to learn flow field behavior. Using a trained model to predict new particle trajectories offers a fixed-small memory footprint and fast inference. To demonstrate and evaluate the proposed method, we perform an in-depth study of performance using a well-known analytical data set, the Double Gyre. Our study considers two flow map extraction strategies as well as the impact of the number of training samples and integration durations on efficacy, evaluates multiple sampling options for training and testing, and informs hyperparameter settings. Overall, we find our method requires a fixed-memory footprint of 10.5 MB to encode a Lagrangian representation of a time-varying vector field while maintaining accuracy. For post hoc analysis, loading the trained model costs only two seconds, significantly reducing the burden of I/O when reading data for visualization. Moreover, our parallel implementation can infer one hundred locations for each of two thousand new pathlines across the entire temporal resolution in 1.3 seconds using one NVIDIA Titan RTX GPU.

KEY WORDS: *Lagrangian Representation, Flow Visualization, Deep Learning*

1. INTRODUCTION

Numerical flow visualization plays a critical role in enabling scientists to understand fluid phenomena and improve computational fluid dynamics models. Although simulations typically produce time-varying vector fields, analysis and visualization are often limited to single time slices due to I/O constraints and memory requirements. Performing accurate time-varying flow visualization using traditional methods requires a high temporal resolution of the vector field data. A potential solution to perform accurate time-varying flow visualization is to consider a Lagrangian representation of the vector field. Lagrangian representations have been demonstrated to offer strong accuracy-storage propositions compared to traditional techniques (Agranovsky

1065–3090/17/$35.00

et al. (2014); Sane et al. (2021a)). The approach involves two phases: in situ and post hoc. Lagrangian representations are extracted from computational simulations using in situ processing and explored during post hoc analysis. In this paper, we study the use of deep learning methods to perform post hoc exploration of time-varying vector fields using reduced Lagrangian representations computed in situ as training data.

In recent years, the scientific visualization community has seen an increased adoption of deep learning (Berger et al. (2018); Engel and Ropinski (2020); Han and Wang (2019); Han et al. (2020); He et al. (2019); Hong et al. (2019); Leventhal et al. (2019); Weiss et al. (2019)), including multiple research projects that consider vector field data (Guo et al. (2020); Han et al. (2018, 2019); Jakob et al. (2020); Kim et al. (2019); Liu et al. (2019); Sahoo and Berger (2021)). With respect to exploratory Lagrangian-based particle advection schemes, the use of deep learning has not previously been studied to the best of our knowledge. Prior strategies have relied on constructing search structures over the data to identify sets of precomputed particle trajectories that can be interpolated across intervals of time. Search structures such as k-d trees and Delaunay triangulations can be computationally expensive to compute for each interval and memory intensive for large data sets (Chandler et al. (2014); Hlawatsch et al. (2010); Sane et al. (2019)). Our study shows that, by leveraging deep learning, we can limit the memory footprint of the extracted data. Importantly, once the model is trained, it provides quick inference of new particle trajectories during post hoc analysis and exploration.

Overall, we contribute the first deep neural network-based method to encode Lagrangian flow maps and enable exploratory particle tracing in time-varying flow fields. Our study demonstrates the performance of the method across varying hyperparameter settings as well as multiple Lagrangian representation configurations. Our trained model requires a fixed-memory footprint of 10.5 MB, potentially offering a potentially significant data reduction for high-resolution flow maps and alleviating I/O costs during exploration. Further, the trained model can infer new trajectories accurately and at rates supporting interactive exploration. Lastly, we consider a widely studied analytical data set, the Double Gyre, as well as, a second vector field targeted to machine learning applications to demonstrate our approach.

2. RELATED WORK

This section provides background on Lagrangian analysis, the use of reduced Lagrangian representations, and the use of machine learning for flow visualization tasks.

2.1 Lagragian Analysis

Lagrangian analysis is a powerful tool, widely adopted by the ocean modeling community (Van Sebille et al. (2018)), to explore time-varying vector fields generated by simulations. In response to growing data set sizes, reduced Lagrangian representations have been increasingly researched as a solution to enable time-varying vector field exploration across various application domains. Reduced Lagrangian representations are computed using in situ processing and explored during post hoc analysis. By utilizing in situ processing, Lagrangian representations are computed using the complete spatial and temporal resolution of the simulation data. Studies have demonstrated reduced Lagrangian representations offer strong accuracy-storage propositions for exploration in temporally sparse settings (Agranovsky et al. (2014); Rapp et al. (2019); Sane et al. (2021a)) as well as directly support feature extraction (Froyland and Junge (2018);

Froyland and Padberg-Gehle (2015); Hadjighasem et al. (2017); Jakob et al. (2020); Schlueter-Kuck and Dabiri (2017)). Additionally, previous research has demonstrated the traditional Eulerian paradigm performs poorly in under-resolved temporal settings (Agranovsky et al. (2014); Da Costa and Blanke (2004); Qin et al. (2014); Rockwood et al. (2019); Sane et al. (2018, 2021a)).

In the Lagrangian specification of a time-varying vector field, information is encoded using particle trajectories. Thus, the Lagrangian representation consists of a collection of particle trajectories spanning the spatial domain and can be defined as a flow map. The flow map $F_{t_0}^{t}(x_0) : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^d$ describes where a massless particle starting at position $x_0 \in \mathbb{R}^d$ and time $t_0 \in \mathbb{R}$ moves in the time interval $[t_0, t] \subset \mathbb{R}$ (Garth et al. (2007)).

Research related to reduced Lagrangian representations that enable time-varying vector fields has advanced along multiple axes. These include in situ sampling techniques (Agranovsky et al. (2014); Rapp et al. (2019); Sane et al. (2019, 2021b)), post hoc reconstruction strategies (Agranovsky et al. (2015); Bujack and Joy (2015); Chandler et al. (2014); Hlawatsch et al. (2010)), theoretical and empirical error analysis (Chandler et al. (2016); Hummel et al. (2016); Sane et al. (2018)), feature extraction (Froyland and Junge (2018); Froyland and Padberg-Gehle (2015); Hadjighasem et al. (2017); Jakob et al. (2020); Schlueter-Kuck and Dabiri (2017)), and application to various domains (Nardini et al. (2017); Sane et al. (2021a); Siegfried et al. (2019)). In this paper, we study the use of deep learning to perform post hoc reconstruction. Specifically, we propose and evaluate the use of multi-layer perceptrons (MLPs) to learn the time-varying vector field behavior from previously computed particle trajectories. With deep learning, a model can be trained once and then be interactively queried at the time of exploration without the significant memory requirements of prior approaches. Our study focuses on the impact of various hyperparameters and extraction configurations on the efficacy of post hoc reconstruction as well as the overall computational cost.

2.2 Flow Visualization Using Machine Learning

In recent years, machine learning techniques have been increasingly researched by the fluid dynamics community (Brunton et al. (2020)). Similarly, with respect to scientific visualization, specifically, flow visualization, the use of machine learning to perform several tasks has increased. For example, it has been widely used to detect flow field features such as eddies and vortices (Bai et al. (2019); Deng et al. (2019); Duo et al. (2019); Lguensat et al. (2018); Liu et al. (2019); Ströfer et al. (2018); Wang et al. (2021); Yi (2018)). Kim and Günther (2019) utilized the convolutional neural networks (CNNs) to extract a robust frame of reference for unsteady two-dimensional (2D) vector fields. Hong et al. (2018) used the long short-term memory (LSTM) to improve data access patterns for improved computational performance during distributed memory particle advection. Li et al. (2015) employed the support vector machine (SVM) to segment streamlines based on user-identified features. For the widely studied task of selecting a representative set of particle trajectories (Sane et al. (2020)), recent state-of-the-art techniques by Han et al. (2018) and Lee and Park (2021) have used deep-learning-based clustering approaches. Further, modern techniques to reconstruct steady state vector fields using a set of streamlines employ machine learning (Han et al. (2019); Sahoo and Berger (2021)).

Jakob et al. (2020) upsampled 2D finite-time lyapunov exponent (FTLE) scalar fields derived from Lagrangian flow maps using an efficient subpixel convolutional neural network (ESPCN) by Shi et al. (2016) and SRCNN by Dong et al. (2015). In our study, we use the Lagrangian representations of 2D time-varying vector fields as data to train neural networks built with MLPs.

We then infer new particle trajectories from the model to support the exploration use case. Our study shows that the application of deep learning to particle tracing can offer the significant benefits of reduced memory requirement and accurate trajectory inference.

3. LAGRANGIAN ANALYSIS USING DEEP LEARNING

We designed our network to learn the flow behavior encoded by the Lagrangian representation of the time-varying vector field. Figure 1a shows the workflow of in situ training data generation process, network training process, and the post hoc inference process. In the in situ extraction phase, Lagrangian flow maps are computed by advecting particles using the full spatial and temporal resolution of the time-varying vector field. We considered two approaches to extract flow maps,

- $Lagrangian_{long}$: extract a single flow map consisting of long particle trajectories with a uniform temporal sampling of each integral curve.

- $Lagrangian_{short}$: extract multiple short flow maps with each flow map consisting of a set of seed locations and a set of end locations for each seed, where each end location in a set corresponds to the displacement from the seed location over non-overlapping intervals of time.

In our paper, we follow the notation used by Agranovsky et al. (2014). We refer to the cycles where the end location is saved out as *file cycles*.

To begin the post hoc analysis phase, the network fetches flow maps from the database, preprocesses them, and loads data as training samples (Section 3.1). The network architecture is built with MLP that are a series of fully connected layers (Section 3.2). The loss function is set to the L1 loss, which is calculated as the error between the target end location and the predicted end location. During the training process, the model takes two parameters, particle start locations and queried file cycles as inputs, and outputs the corresponding end locations. Weights of the model are updated by backpropagation of the loss to find the optimized weights (Section 3.3). Finally, new trajectories can be infered from the trained model (Section 3.4).

3.1 Training Data Generation

We stored extracted Lagrangian flow maps in the form of training data for the model. We considered two strategies to sample the time-varying vector field. The first strategy, $Lagrangian_{long}$, involves computing long trajectories with uniform sampling along the curve. Reconstruction of new trajectories using long precomputed trajectories is more accurate when the propagation of error is eliminated after every interpolation step (Hummel et al. (2016); Sane et al. (2019)). However, the quality of domain coverage may be reduced as the integration time increases due to divergence in the flow field (Chandler et al. (2016)). The second strategy, $Lagrangian_{short}$, involves computing sets of short trajectories with only the start and end location after non-overlapping intervals of time stored. Although such an approach offers improved domain coverage (Agranovsky et al. (2014)), the particle trajectory reconstruction may be less accurate due to error propagation (Bujack and Joy (2015)).

For both approaches, the first step is placing sample seeds in the domain. In this paper, we denote the number of seeds by N. To understand the impact of the seed placement strategy on the model inference performance, we studied three strategies: (1) seeding along a uniform grid

($uniform$), (2) seeding using a pseudorandom number sequence ($random$), and (3) seeding using a Sobol quasirandom sequence ($sobol$). Specifically, we considered reconstruction accuracy near features of interest and boundaries. Although placing uniform seeds can provide good domain coverage and fast interpolation during post hoc analysis, it does not optimize information per byte stored. Thus, in many practical cases, the Lagrangian representation can be unstructured and would typically incur a higher interpolation cost during post hoc analysis. By considering $random$ and $sobol$ seeding, we were able to demonstrate the fast inference of new trajectories from unstructured Lagrangian flow maps. We compare these three seeding choices in Section 4.2.1.

After seeds are placed, particle trajectories are computed by displacing particles from time t to $t + \delta$, where δ indicates an advancement by one simulation time step. Following the notation in Agranovsky et al. (2014), we refer to one simulation advancement as a *cycle*, the cycle on which the simulation saves data as a *file cycle*, and the number of cycles between file cycles as the *interval* in the following sections. Given a total temporal duration T, the total number of file cycles n can be calculated by

$$n = floor(T/(\delta * C)) \tag{1}$$

where C represents the file cycle *interval*. Thus, the list of file cycles is $C_{0:n-1} = [C, 2C, 3C, ..., nC]$. To generate $Lagrangian_{long}$ flow maps, seeds are placed once at the beginning at time $t_0 = 0$ and traced until T, i.e., the entire temporal duration. Intermediate locations are recorded along each trajectory at every file cycle. To generate $Lagrangian_{short}$ flow maps, particle tracing starts at time $t_0 = 0$ and terminates at time $t_1 = t_0 + \delta * C$. Then, the location at t_1 is saved, and seeds are reset for the tracing until the next file cycle. This process is repeated until the last file cycle.

The training data sets are saved in the NPY file format for efficient loading in Python. We created a three-dimentional (3D) array, with dimensions of $[n + 1, N, 3]$, for saving start seed locations and corresponding end locations at various file cycles. When loading the data sets, the data are organized into training samples, as shown in Equation 2. One training sample contains start location $start_i$ (where $i = 0, 1, ...N-1$), the queried file cycle C_j (where $j = 0, 1, ...n-1$), and the target end location at the queried file cycle $target_{i,j}$ (where $i = 0, 1, ...N - 1$ and $j = 0, 1, ...n - 1$). The start location and the queried file cycle are inputs to the network. The target end locations are used for calculating the loss (Equation 3). In addition to training data, we generated validation data by using $0.1 * N$ seeds (10% of training samples) and following the same process.

$$\begin{aligned} Inputs = &\{\{start_0, C_0, target_{0,C_0}\}, \\ &\{start_0, C_1, target_{0,C_1}\}, ..., \\ &\{start_0, C_{n-1}, target_{0,C_{n-1}}\}, ..., \\ &\{start_{N-1}, C_{n-1}, target_{N-1,C_{n-1}},\}\} \end{aligned} \tag{2}$$

3.2 Network Architecture

The network architecture, shown in Figure 1b, consists of a latent encoder E and a latent decoder D. The latent encoder E and decoder D are built with MLP, a series of fully connected layers. The latent encoder E takes a particle's start location $start$ and a queried file cycle C_j as inputs. These two parameters are separately fed into two sequences of fully connected layers of size (64, 128, 256, 512) and (16, 32, 64, 128, 256, 512). The two outputs are then concatenated together

as a latent vector. Next, the latent decoder D that is also a series of fully connected layers of size (512, 256, 128, 64) is followed by the latent vector being mapped to predicted end location $pred$ at the queried file cycle. We added layer normalization (Ba et al. (2016)) after each fully connected layer except output layers to stabilize the training process. Moreover, we used the rectified linear unit (ReLU) (Nair and Hinton (2010)) as the activation function for each output from the fully connected layer.

3.3 Training Process

Algorithm 1: Training Process

Input: Data set shown in Equation 2
Initial weights of the network w
Output: Optimized weights w
Load training data set
for *each epoch* **do**
  for *each batch of training samples* **do**
    model.train()
    $pred = model(start, queried\_file\_cycle)$
    $loss = L1\_Loss(pred, target)$
    Backpropagation and update weight w
  end
  for *each batch of validation samples* **do**
    model.eval()
    $pred = model(start, queried\_file\_cycle)$
    $loss = L1\_Loss(pred, target)$
  end
  call learning rate scheduler adjust the learning rate if needed
end

We implemented our neural network using Pytorch (Paszke et al. (2019)). The training process, shown in Algorithm 1, aims to find the optimized weights w of the network. The weights are initialized by Pytorch. We created a custom Pytorch *Dataset* class to load and store all training samples. We then loaded the Pytorch *Dataset* object into a Pytorch *DataLoader* for iterating through the training samples. At the beginning of each epoch, the training samples are shuffled and split into batches. Given a batch of training samples, the forward process computes the output following the network architecture and computes the loss as defined by the loss function. The backpropagation process is done automatically using Pytorch by calling *loss.backward()*, and the weights are updated by the optimizer. For our experiments, we trained the network for 100 epochs using the Adam optimizer (Kingma and Ba (2014)) with the hyperparameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1e-6$. Further, in our training process, we set the initial learning rate to 10^{-5} and used a learning rate scheduler (Contributors (2019)), provided by Pytorch to reduce the current learning rate by a factor of 2 if the validation loss had not decreased for five epochs. We applied L1 loss as loss functions in our method. **L1 loss** calculates the mean absolute error between target and predicted end locations by the network (Equation 3).

$$L1\_Loss = |target - pred| \tag{3}$$

3.4 Inference Process

Besides varying generation processes for $Lagrangian_{long}$ and $Lagrangian_{short}$, the inference process when using the model trained by data from these two approaches also varies. When using $Lagrangian_{long}$, interpolations are performed by always considering the new seed start location at $t_0 = 0$. The end location inferred by the model results from the provided start location and the queried file cycle. In contrast, when using $Lagrangian_{short}$, new particle trajectories are "stitched" together by advancing the new seed across intervals. Here, the inference is performed by considering the location of the seed particle at the previous file cycle and the target file cycle. Since every inference except the first uses previously inferred results, errors might propagate along new trajectories when using $Lagrangian_{short}$ (Hummel et al. (2016); Sane et al. (2019)). We refer to the absolute error introduced by the model for any single inference as *local error* and to the error accumulated along particle trajectories that are "stitched" together as *global error*. Similar to other Lagrangian-based advection schemes, our inference process currently is limited to interpolating the locations along a particle trajectory at file cycles, and in the case of $Lagrangian_{long}$, it is limited to particles starting at $t_0 = 0$.

To measure the accuracy of new particle trajectories inferred by the model, we calculated the average of aggregated Euclidean distance between the target ground truth and the model predicted along each trajectory (Equation 4).

$$error_i = \frac{1}{n} \sum_{j=0}^{n} L_2(target_j - pred_j) \tag{4}$$

where i represents the index of the new seed and n is the number of end locations (file cycles) along the trajectories (Equation 1).

4. RESULTS

In this section, we first describe the data set used for our experiments (Section 4.1). Next, we present an evaluation of sampling strategies and hyperparameters (learning rate, batch size) used during training data generation (Section 4.2), followed by a report of the performance of our proposed network for training and inferences (Section 4.3). Finally, to evaluate the accuracy of the model across Lagrangian flow map extraction parameter settings, we quantitatively and qualitatively evaluate the impact of varying the number of seeds (Section 4.4) and file cycle intervals (Section 4.5).

4.1 Data Set

We conducted our study by considering a standard benchmark data set frequently used to study fluid dynamics and, in particular, flow visualization tools and techniques: the 2D unsteady Double Gyre Shadden et al. (2005). The model of the unsteady Double Gyre flow field is widely studied for the computation of hyperbolic Lagrangian coherent structures (LCS) in flow data. For all the training data generated, we considered a total temporal duration of $[0, 10]$ with $\delta = 0.01$.

The Double Gyre flow field is defined by equation 5 within the spatial domain $[0, 2] \times [0, 1]$.

$$\begin{aligned} \psi(x, y, t) &= A sin(\pi f(x, t)) sin(\pi y) \\ f(x, t) &= a(t)x^2 + b(t)x \\ a(t) &= \varepsilon sin(\omega t) \\ b(t) &= 1 - 2\varepsilon sin(\omega t) \\ \text{where } A &= 0.1,\ \omega = pi/5 \text{ and } \varepsilon = 0.25 \end{aligned} \tag{5}$$

Our training data generation process used the analytical solution (Equation 5) for particle advection during Lagrangian flow map computation. We show the velocity field at time 0 (Figure 2a) and the FTLE (Figure 2b) of the Double Gyre data set. The ridges of the FTLE scalar field are used to approximate Lagrangian Coherent Structures in the flow. We extended the 2D Double Gyre data sets to 3D by adding the same z-axis to every seed. The size of training data sets increases linearly with a larger number of seeds and shorter intervals. In our experiments, the minimum and maximum sizes of the reduced Lagrangian representation training data were 2.6MB and 24.2MB, respectively. We did not observe significant improvements of accuracy using more training data for this data set. We generated all the training data sets using a desktop equipped with an Intel(R) Xeon(R) W-3275M CPU (56 cores; 256GB memory) and one NVIDIA Titan RTX GPU. We computed the particle trajectories of the Lagrangian flow maps in parallel using the TBB library (Intel (2007)).

4.2 Evaluation of Seeding Strategy and Hyperparameters Settings

Our model was implemented using the Pytorch library (Paszke et al. (2019)) and trained on dual RTX 3090s GPUs. We considered two methods of extracting training data sets (Section 3.1): $Lagrangian_{long}$ and $Lagrangian_{short}$. We studied the impact of seeding strategy as well as the learning rate and batch size for each flow map extraction approach.

4.2.1 Seeding Strategy

To generate training data, we evaluated three seed placement strategies: (1) seeding along a uniform grid ($uniform$), (2) seeding using a pseudorandom number sequence ($random$), and (3) seeding using a Sobol quasirandom sequence ($sobol$). For this experiment, we sampled the time-varying Double Gyre vector field domain using 2,000 seeds and a fixed file cycle interval of 30. All models were trained with a batch size of 200 and a learning rate of 0.001. For the uniform sampling experiment, we used a $[50 \times 40]$ grid. Further, besides applying these three seed placement strategies to generate training data sets, we also considered the strategies for testing seeds. Figure 3 presents error maps produced by various combinations of seed placement strategies for training and testing data, as well as outcomes considering two flow map extractions strategies. Comparing error maps evaluated by using $Lagrangian_{long}$ for sampling time-varying vector field (Figure 3a), we found that the Sobol quasirandom sequence ($sobol$) was slightly better than the pseudorandom number sequence ($random$). They both produced more accurate results for the testing seeds that were not on the boundary. The uniform seeding ($uniform$) was more accurate only when the testing seeds were also uniform. Moreover, the Sobol quasirandom sequence ($sobol$) performed better than the pseudorandom number sequence ($random$) when sampling the time-varying vector field using $Lagrangian_{short}$, and they were both better than the uniform seeding ($uniform$) (Figure 3b) except for seeds on the boundary. We chose the Sobol

quasirandom sequence ($sobol$) as the seeding strategy in all our following experiments. Further work is required to identify sampling strategies that optimize the quality of the training data.

4.2.2 Learning Rate and Batch Size

The learning rate is a critical hyperparameter for a deep neural network. We examined four settings of the learning rate: 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5} for $Lagrangian_{long}$ and $Lagrangian_{short}$. For all experiments, the training data sets were generated with 5,000 seeds and a file cycle interval of 30 using the $sobol$ seed placement method with the Double Gyre data set. The batch size was set to 200. The learning rate of 10^{-2} resulted in the model failing to converge; therefore, we did not use it for comparison. We found the learning rates of 10^{-3} and 10^{-4} were better for our model when the training data sets were generated using the $Lagrangian_{long}$ flow map extraction strategy (Figure 4(a)). The learning rates of 10^{-3}, 10^{-4}, and 10^{-5} resulted in a similar loss when the model was trained using data sets generated using the $Lagrangian_{short}$ approach (Figure 4(b)).

To identify the optimal combination of batch size with the learning rates of 10^{-3} and 10^{-4}, we conducted a set of experiments. Our experiments considered three options for batch size, two options for total number of training samples, and both flow map extraction strategies ($Lagrangian_{long}$ and $Lagrangian_{short}$). Figure 5) presents violin plots of the error for reconstructed trajectories. Although we found the choice of learning rate and flow map extraction strategy could significantly impact accuracy, varying the batch size did not result in a significant change of accuracy for a fixed learning rate and flow extraction strategy.

4.3 Network Training and Inference

Table 1 reports time spent training the model, memory consumption for saving the trained model, and the inference time to generate new trajectories with the trained model. As expected, the training time increased linearly with the number of training samples for both approaches. The storage cost for saving the trained model, irrespective of the data set or number of training samples, was fixed. Based on the network's parameters, the trained models required the same memory size of 10.5 MB. However, verification as well as understanding the impacts of complex, turbulent, and 3D flow fields on network training and performance requires future in-depth investigation. That said, considering the network's parameters are independent of the complexity of the flow field, we expect our method to scale and be used to reduce the memory footprint of large-scale high-resolution Lagrangian representations of time-varying vector fields. An important consequence of a small memory footprint is the reduced cost of two seconds to load the entire model, thus alleviating the system from expensive I/O for loading data during exploratory visualization. Further, our results show parallel inference of 2,000 trajectory with 20 locations interpolated to approximate each curve costs 0.38s using the same machine as for generating training data sets.

4.4 Impact of Number of Seeds

We evaluated the impact of the number of seeds on the performance of our model qualitatively and quantitatively. We used a fixed file cycle interval of 30 for all training data discussed in this section. We created training data sets with four options for number of seeds, 5,000, 10,000, 15,000, and 20,000, for the $Lagrangian_{long}$ and $Lagrangian_{short}$ approaches. To evaluate the accuracy of the reconstruction, 2,000 random particles were seeded in the domain. To avoid

extrapolation errors due to our use of the *sobol* seeding strategy for training data generation (Section 3.1), we used a boundary offset of 0.05 to prevent test seeds from being placed exactly on the boundary.

In Figure 6, we report the error map as well as the FTLE derived from using various configurations for training data generation. The result highlighted the relation of the trained model's performance and flow features in the domain. The error for each trajectory was measured using Equation 4. We observed reconstruction errors were higher in regions with greater separation in the flow field, i.e., regions with higher FTLE values. Moreover, for both $Lagrangian_{long}$ and $Lagrangian_{short}$, the error maps confirmed that increasing the number of seeds could increase the inference accuracy. In addition, we visualized the distribution of errors for the model-generated results in comparison to the ground truth (Figure 7). We observed a decreasing median error as the number of seeds used to sample the domain increased. However, the reduction in error was less after 10,000 seeds. Further, the models trained with $Lagrangian_{short}$ data sets showed greater global error due to local error propagation during reconstruction of new trajectories. In the derived FTLE fields in Figure 2b, although the FTLE ridges are visible in all reconstructions, the $Lagrangian_{long}$ can support accurate reconstruction of the entire field, whereas the $Lagrangian_{short}$ reconstructions produce minor artifacts in regions of low separation.

Finally, to assess the inference results qualitatively, Figure 8 shows the model-generated trajectories and the ground truth Double Gyre trajectories by varying the number of training seeds. The reconstructed results were almost identical to the ground truth for all new trajectories when 10,000 or more seeds were used for training. When 5,000 seeds were used for training, the $Lagrangian_{short}$ demonstrated lower reconstruction accuracy as interpolation error propagates and accumulates. In contrast, the $Lagrangian_{long}$ closely followed the ground truth. Here, each location along the trajectory was interpolated directly from the starting seed location. For the $Lagrangian_{long}$, even training data generated using 5,000 seeds were sufficient to maintain accuracy.

4.5 Impact of File Cycle Interval

To understand the performance of our model with varying file cycle intervals, we evaluated four intervals, 10, 20, 50, and 100, in our experiments. We considered a total of 1,000 cycles of the Double Gyre data set. Further, we used a fixed number of 10,000 seeds to generate the training data sets.

In Figure 9, we report the error maps as well as the FTLE derived from using various configurations for training data generation. The $Lagrangian_{long}$ was not impacted by the file cycle interval since each interpolation was independent of prior locations stored along the trajectory. Reconstruction of new trajectories using the model trained by the $Lagrangian_{short}$ data involved an interpolation process where each location along the trajectory was dependent on the previous location. Thus, we observed a higher reconstruction error when the interval was short, and more intervals need to be spanned to construct a trajectory over the entire temporal duration. For example, for training data generated by the $Lagrangian_{short}$ using an interval of 10, we saw the reconstruction error was higher for particles originating near FTLE ridges. These findings are consistent with the error analysis of Lagrangian-based particle tracing systems (Chandler et al. (2016)). Similar to prior experiments, in Figure 2b, we observed the derived FTLE scalar fields are accurate for the $Lagrangian_{long}$, but contained some artifacts for the $Lagrangian_{short}$. Here, as expected, the $Lagrangian_{short}$ shows fewer artifacts when using a longer file cycle interval.

Considering the violin plots in Figure 10, we obsersed varying reconstruction accuracy patterns. The $Lagrangian_{long}$ accuracy did not change significantly with the file cycle interval. The local error of the $Lagrangian_{short}$ was low for short intervals, but increased as the interval length increased due to greater divergence between neighboring trajectories over longer integration times. The global error of the $Lagrangian_{short}$ represented the accuracy of particle trajectories that are "stitched". We found the global error was the highest when the file cycle interval was short given a greater number of "stitching" events were involved. As the file cycle interval increased, although the accuracy of every individual interpolation (local error) was higher, the global error decreased due to fewer total interpolation steps. Again, these findings are consistent with prior work by Chandler et al. (2016) and Sane et al. (2019). Additionally, we present the average error across all particles over time for the $Lagrangian_{long}$ and $Lagrangian_{short}$ approaches in Figure 11. The line curves provide strong evidence of local error propagation and accumulation for tests using $Lagrangian_{short}$ training data.

For a qualitative assessment of the impact of the file cycle interval, we present reconstructed pathlines alongside the ground truth in Figure 12. We used piecewise linear interpolation to connect every interpolated location along the new trajectories. Although the $Lagrangian_{short}$ demonstrated a small deviation from the ground truth when short file cycle intervals were used, the overall accuracy of reconstructed trajectories was high with interpolated results closely overlapping the ground truth.

4.6 Application to Fluid Dynamics Machine Learning Data Set

We applied our method to an ensemble member (#200) of the 2D fluid dynamics machine learning data set generated using the Gerris flow solver (Jakob et al. (2020)). The resolution of the original data set is $[512 \times 512 \times 1001]$. To generate the training data set, we placed $50,000$ seeds in the domain, set the file cycle interval to 10, and traced flow maps over the first 100 cycles. For particle advection, we used the VTK-m (Moreland et al. (2016)) library and a fourth-order Runge-Kutta (RK4) advection kernal. The median error of using our method after 100 cycles and 10 interpolation steps is approximately two times the grid cell size. Our method cost 0.6 seconds for reconstructing 2,000 particle trajectories using parallel inferences with OpenMP (Dagum and Menon (1998)). When considering the storage requirements, the subset of the original data size we consider is approximately 209MB. Since our model has a fixed memory requirement, once trained, the storage costs are still fixed at 10.5 MB. To qualitatively evaluate the reconstructed data, we visualize pathlines inferred by the trained model in comparison with the ground truth in Figure 13. In future works, we aim to study how to improve inteprolation accuracy as well as determine an appropriate number of samples to be computed using in situ processing.

5. FUTURE WORK AND CONCLUSION

Exploratory flow visualization for large-scale time-varying vector field data is challenging. In this paper, we introduced a deep neural network-based approach using Lagrangian represesntations to enable exploratory analysis. Our study demonstrated our model can be trained using Lagrangian representations extracted from a 2D time-varying vector field. Specifically, we used the widely studied unsteady Double Gyre analytical flow data set and one fluid dynamics machine learning data set to demonstrate our method. We contributed the first assessment of applying deep learning to various forms of Lagrangian representations and evaluated the efficacy of exploratory analysis. A benefit of using our method is the fixed memory required by a model and

fast inference of unstructured spatiotemporal data. Our trained model requires only 10.5 MB, and consequently, time spent on I/O to load the model during post hoc analysis is negligible. Further, we are able to infer the pathlines of thousands of particles at interactive rates. With respect to reconstruction interpolation error, we found inference errors are small and follow predictable patterns consistent with results from prior works. Predictable and consistent error patterns enable effective future navigation of strategies to reduce reconstruction interpolation error when using machine learning. Overall, our study demonstrates the benefits of leveraging deep learning for exploratory flow visualization of time-varying vector field data.

An important direction for future work is investigating model performance for more complex or turbulent flows as well as large-scale 3D flow fields. With the objectives of improving spatial and temporal interpolation accuracy and reducing model training time, various forms of training data to train a model or different network architectures could be considered. For example, concatenate sets of $Lagrangian_{long}$ trajectories to limit instances of error propagation while simultaneously accounting for reduced interpolation error due to stretching or divergence in the flow. Lastly, an open-source interactive tool for interactive flow visualization exploration, with a trained model serving as a backend, would be valuable to the community. We plan to pursue these projects in the future.

ACKNOWLEDGMENTS

The authors acknowledge current research support provided in part by the Intel Graphics and Visualization Institutes of XeLLENCE, the National Institutes of Health under grant numbers P41 GM103545 and R24 GM136986, the Department of Energy under grant number DE-FE0031880, and the Utah Office of Energy Development.

REFERENCES

Agranovsky, A., Camp, D., Garth, C., Bethel, E.W., Joy, K.I., and Childs, H., Improved Post Hoc Flow Analysis Via Lagrangian Representations, *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 67–75, 2014.

Agranovsky, A., Obermaier, H., Garth, C., and Joy, K.I., A Multi-Resolution Interpolation Scheme for Pathline Based Lagrangian Flow Representations, *Visualization and Data Analysis 2015*, Vol. 9397, p. 93970K, 2015.

Ba, J.L., Kiros, J.R., and Hinton, G.E., Layer Normalization, *arXiv preprint arXiv:1607.06450*, 2016. URL `https://arxiv.org/pdf/1607.06450.pdf`

Bai, X., Wang, C., and Li, C., A Streampath-Based RCNN Approach to Ocean Eddy Detection, *IEEE Access*, vol. **7**, pp. 106336–106345, 2019.

Berger, M., Li, J., and Levine, J.A., A Generative Model for Volume Rendering, *IEEE transactions on visualization and computer graphics*, vol. **25**, no. 4, pp. 1636–1650, 2018.

Brunton, S.L., Noack, B.R., and Koumoutsakos, P., Machine Learning for Fluid Mechanics, *Annual Review of Fluid Mechanics*, vol. **52**, pp. 477–508, 2020.

Bujack, R. and Joy, K.I., Lagrangian Representations of Flow Fields with Parameter Curves, *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, IEEE, pp. 41–48, 2015.

Chandler, J., Bujack, R., and Joy, K.I., Analysis of Error in Interpolation-Based Pathline Tracing, *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pp. 1–5, 2016.

Chandler, J., Obermaier, H., and Joy, K.I., Interpolation-Based Pathline Tracing in Particle-Based Flow

REFERENCES

Agranovsky, A., Camp, D., Garth, C., Bethel, E.W., Joy, K.I., and Childs, H., Improved Post Hoc Flow Analysis Via Lagrangian Representations, 2*014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 67–75, 2014.

Agranovsky, A., Obermaier, H., Garth, C., and Joy, K.I., A Multi-Resolution Interpolation Scheme for Pathline Based Lagrangian Flow Representations, *Visualization and Data Analysis 2015*, Vol. 9397, p. 93970K, 2015.

Ba, J.L., Kiros, J.R., and Hinton, G.E., Layer Normalization, arXiv preprint arXiv:1607.06450, 2016.URL https://arxiv.org/pdf/1607.06450.pdf

Bai, X., Wang, C., and Li, C., A Streampath-Based RCNN Approach to Ocean Eddy Detection, *IEEE Access*, vol. **7**, pp. 106336–106345, 2019.

Berger, M., Li, J., and Levine, J.A., A Generative Model for Volume Rendering, IEEE Transactions on Visualization Aand Computer Graphics, vol. **25**, no. 4, pp. 1636–1650, 2018.

Brunton, S.L., Noack, B.R., and Koumoutsakos, P., Machine Learning for Fluid Mechanics, *Annual Review of Fluid Mechanics*, vol. **52**, pp. 477–508, 2020.

Bujack, R. and Joy, K.I., Lagrangian Representations of Flow Fields with Parameter Curves, *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, IEEE, pp. 41–48, 2015.

Chandler, J., Bujack, R., and Joy, K.I., Analysis of Error in Interpolation-Based Pathline Tracing, Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers, pp. 1–5, 2016.

Chandler, J., Obermaier, H., and Joy, K.I., Interpolation-Based Pathline Tracing in Particle-

Based Flow Visualization, IEEE Transactions on Visualization and Computer Graphics, vol. **21**, no. 1, pp. 68–80, 2014.

Contributors, T., Learning Rate Scheduler, , 2019.URL https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

Da Costa, M.V. and Blanke, B., Lagrangian Methods for Flow Climatologies and Trajectory Error Assessment, *Ocean Modelling*, vol. **6**, no. 3-4, pp. 335–358, 2004.

Dagum, L. and Menon, R., OpenMP: An Industry Standard API for Shared-Memory Programming, *IEEE Computational Science and Engineering*, vol. **5**, no. 1, pp. 46–55, 1998.

Deng, L., Wang, Y., Liu, Y., Wang, F., Li, S., and Liu, J., A CNN-Based Vortex Identification Method, *Journal of Visualization*, vol. **22**, no. 1, pp. 65–78, 2019.

Dong, C., Loy, C.C., He, K., and Tang, X., Image Super-Resolution Using Deep Convolutional Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. **38**, no. 2, pp. 295–307, 2015.

Duo, Z., Wang, W., and Wang, H., Oceanic Mesoscale Eddy Detection Method Based on Deep Learning, Remote Sensing, vol. **11**, no. 16, p. 1921, 2019.

Engel, D. and Ropinski, T., Deep Volumetric Ambient Occlusion, *IEEE Transactions on Visualization and Computer Graphics*, vol. **27**, no. 2, pp. 1268–1278, 2020.

Froyland, G. and Junge, O., Robust FEM-Based Extraction of Finite-Time Coherent Sets Using Scattered, Sparse, and Incomplete Trajectories, *SIAM Journal on Applied Dynamical Systems*, vol. **17**, no. 2, pp. 1891–1924, 2018.

Froyland, G. and Padberg-Gehle, K., A Rough-and-Ready Cluster-Based Approach for Extracting Finite-Time Coherent Sets from Sparse and Incomplete Trajectory Data, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. **25**, no. 8, p. 087406, 2015.

Garth, C., Gerhardt, F., Tricoche, X., and Hans, H., Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications, *IEEE Transactions on Visualization and Computer Graphics*, vol. **13**, no. 6, pp. 1464–1471, 2007.

Guo, L., Ye, S., Han, J., Zheng, H., Gao, H., Chen, D.Z., Wang, J.X., and Wang, C., SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization, 2020 IEEE Pacific Visualization Symposium (PacificVis), *IEEE Computer Society*, pp. 71–80, 2020.

Hadjighasem, A., Farazmand, M., Blazevski, D., Froyland, G., and Haller, G., A Critical Comparison of Lagrangian Methods for Coherent Structure Detection, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. **27**, no. 5, p. 053104, 2017.

Han, J., Tao, J., and Wang, C., FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces, *IEEE Transactions on Visualization and Computer Graphics*, vol. **26**, no. 4, pp. 1732–1744, 2018.

Han, J., Tao, J., Zheng, H., Guo, H., Chen, D.Z., and Wang, C., Flow Field Reduction via Reconstructing Vector Data from 3-D Streamlines Using Deep Learning, *IEEE Computer Graphics and Applications*, vol. **39**, no. 4, pp. 54–67, 2019.

Han, J. and Wang, C., TSR-TVD: Temporal Super-Resolution for Time-Varying Data Analysis and Visualization, *IEEE Transactions on Visualization and Computer Graphics*, vol. **26**, no. 1, pp. 205–215, 2019.

Han, J., Zheng, H., Xing, Y., Chen, D.Z., and Wang, C., V2V: A Deep Learning Approach to Variable-to-Variable Selection and Translation for Multivariate Time-Varying Data, *IEEE Transactions on Visualization and Computer Graphics*, vol. **27**, no. 2, pp. 1290–1300, 2020.

He, W., Wang, J., Guo, H., Wang, K.C., Shen, H.W., Raj, M., Nashed, Y.S., and Peterka, T., InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations,

IEEE Transactions on Visualization and Computer Graphics, vol. **26**, no. 1, pp. 23–33, 2019.

Hlawatsch, M., Sadlo, F., and Weiskopf, D., Hierarchical Line Integration, *IEEE Transactions on Visualization and Computer Graphics*, vol. **17**, no. 8, pp. 1148–1163, 2010.

Hong, F., Liu, C., and Yuan, X., DNN-VolVis: Interactive Volume Visualization Supported by Deep Neural Network, *2019 IEEE Pacific Visualization Symposium (PacificVis)*, *IEEE*, pp. 282–291, 2019.

Hong, F., Zhang, J., and Yuan, X., Access Pattern Learning with Long Short-Term Memory for Parallel Particle Tracing, *2018 IEEE Pacific Visualization Symposium (PacificVis)*, *IEEE*, pp. 76–85, 2018.

Hummel, M., Bujack, R., Joy, K.I., and Garth, C., Error Estimates for Lagrangian Flow Field Representations, *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pp. 7–11, 2016.

Intel, Intel Threading Building Blocks, 2007. URL https://www.intel.com/content/www/us/en/develop/tools/oneapi/components/onetbb.html

Jakob, J., Gross, M., and Günther, T., A Fluid Flow Data Set for Machine Learning and its Application to Neural Flow Map Interpolation, *IEEE Transactions on Visualization and Computer Graphics*, vol. **27**, no. 2, pp. 1279–1289, 2020.

Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B., Deep Fluids: A Generative Network for Parameterized Fluid Simulations, *Computer Graphics Forum*, Vol. 38, Wiley Online Library, pp. 59–70, 2019.

Kim, B. and Günther, T., Robust Reference Frame Extraction from Unsteady 2D Vector Fields with Convolutional Neural Networks, Computer Graphics Forum, Vol. 38, Wiley Online Library, pp. 285–295, 2019.

Kingma, D.P. and Ba, J., Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980, 2014.

Lee, J.Y. and Park, J., Deep Regression Network-Assisted Efficient Streamline Generation Method, *IEEE Access*, vol. **9**, pp. 111704–111717, 2021.

Leventhal, S., Kim, M., and Pugmire, D., PAVE: An In Situ Framework for Scientific Visualization and Machine Learning Coupling, *2019 IEEE/ACM 5th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-5), IEEE*, pp. 8–15, 2019.

Lguensat, R., Sun, M., Fablet, R., Tandeo, P., Mason, E., and Chen, G., EddyNet: A Deep Neural Network for Pixel-Wise Classification of Oceanic Eddies, *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, IEEE*, pp. 1764–1767, 2018.

Li, Y., Wang, C., and Shene, C.K., Extracting Flow Features via Supervised Streamline Segmentation, *Computers & Graphics*, vol. **52**, pp. 79–92, 2015.

Liu, Y., Lu, Y., Wang, Y., Sun, D., Deng, L., Wang, F., and Lei, Y., A CNN-Based Shock Detection Method in Flow Visualization, *Computers & Fluids*, vol. **184**, pp. 1–9, 2019.

Moreland, K., Sewell, C., Usher, W., Lo, L.T., Meredith, J., Pugmire, D., Kress, J., Schroots, H., Ma, K.L., Childs, H., VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures, *IEEE Computer Graphics and Applications*, vol. **36**, no. 3, pp. 48–58, 2016.

Nair, V. and Hinton, G.E., Rectified Linear Units Improve Restricted Boltzmann Machines, *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814, 2010.

Nardini, P., Böttinger, M., Scheuermann, G., and Schmidt, M., Visual Study of the Benguela Upwelling System Using Pathline Predicates, *Proceedings of the Workshop on Visualisation in Environmental Sciences*, pp. 19–23, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems*, vol. **32**, pp. 8026–8037, 2019.

Qin, X., van Sebille, E., and Gupta, A.S., Quantification of Errors Induced by Temporal Resolution on Lagrangian Particles in an Eddy-Resolving Model, *Ocean Modelling*, vol. **76**, pp. 20–30, 2014.

Rapp, T., Peters, C., and Dachsbacher, C., Void-and-Cluster Sampling of Large Scattered Data and Trajectories, Ieee Transactions on Visualization and Computer Graphics, vol. **26**, no. 1, pp. 780–789, 2019.

Rockwood, M.P., Loiselle, T., and Green, M.A., Practical Concerns of Implementing a Finite-Time Lyapunov Exponent Analysis with Under-Resolved Data, *Experiments in Fluids*, vol. **60**, no. 4, pp. 1–16, 2019.

Sahoo, S. and Berger, M., Integration-Aware Vector Field Super Resolution, 2021.

Sane, S., Bujack, R., and Childs, H., Revisiting the Evaluation of In Situ Lagrangian Analysis, EGPGV@ EuroVis, pp. 63–67, 2018.

Sane, S., Bujack, R., Garth, C., and Childs, H., A Survey of Seed Placement and Streamline Selection Techniques, Computer Graphics Forum, Vol. 39, Wiley Online Library, pp. 785–809, 2020.

Sane, S., Childs, H., and Bujack, R., An Interpolation Scheme for VDVP Lagrangian Basis Flows, *Euro- graphics Symposium on Parallel Graphics and Visualization*, pp. 109–119, 2019.

Sane, S., Johnson, C.R., and Childs, H., Investigating In Situ Reduction via Lagrangian Representations for Cosmology and Seismology Applications, *International Conference on Computational Science*, Springer, pp. 436–450, 2021a.

Sane, S., Yenpure, A., Bujack, R., Larsen, M., Moreland, K., Garth, C., Johnson, C.R., and Childs, H., Scalable In Situ Computation of Lagrangian Representations via Local Flow Maps, 2021b.

Schlueter-Kuck, K.L. and Dabiri, J.O., Coherent Structure Colouring: Identification of Coherent Structures from Sparse Data Using Graph Theory, *Journal of Fluid Mechanics*, vol. **811**, pp. 468–486, 2017.

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**, nos. 3-4, pp. 271–304, 2005.

Shi, W., Caballero, J., Husza´r, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., and Wang, Z., Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883, 2016.

Siegfried, L., Schmidt, M., Mohrholz, V., Pogrzeba, H., Nardini, P., Böttinger, M., and Scheuermann, G., The Tropical-Subtropical Coupling in the Southeast Atlantic from the Perspective of the Northern Benguela Upwelling System, *PloS One*, vol. **14**, no. 1, p. e0210083, 2019.

Stro¨ fer, C.M., Wu, J., Xiao, H., and Paterson, E., Data-Driven, Physics-Based Feature Extraction from Fluid Flow Fields Using Convolutional Neural Networks, *Communications in Computational Physics*, vol. **25**, no. 3, pp. 625–650, 2018.

Van Sebille, E., Griffies, S.M., Abernathey, R., Adams, T.P., Berloff, P., Biastoch, A., Blanke, B., Chas- signet, E.P., Cheng, Y., Cotter, C.J., Lagrangian Ocean Analysis: Fundamentals and Practices, *Ocean Modelling*, vol. **121**, pp. 49–75, 2018.

Wang, Y., Deng, L., Yang, Z., Zhao, D., and Wang, F., A Rapid Vortex Identification Method Using Fully Convolutional Segmentation Network, *The Visual Computer*, vol. **37**, no. 2, pp. 261–273, 2021.

Weiss, S., Chu, M., Thuerey, N., and Westermann, R., Volumetric Isosurface Rendering with Deep Learning-Based Super-Resolution, *IEEE Transactions on Visualization and Computer Graphics*, vol. **27**, no. 6, pp. 3064 – 3078, 2019.

Yi, T.B.L., CNN-based Flow Field Feature Visualization Method, *International Journal of Performability Engineering*, vol. **14,** no. 3, p. 434, 2018.

(a) The workflow of our proposed approach. The Lagrangian flow maps are calculated using in situ processing and saved to the database. The network is trained using the particle start locations and the corresponding end locations at various file cycles. Once the model is fully trained, new particle trajectories can be inferred from the model.

(b) The architecture of our neural network built with multi-layer perceptrons (MLP). The network takes the particle start location and the file cycles as input, and outputs the particle end locations.

FIG. 1: Unlike prior two-phase Lagrangian analysis workflows, after extracting Lagrangain representations using in situ processing, a preprocessing phase involving neural network training is introduced prior to post hoc analysis. Figure 1a shows the high-level workflow of our proposed approach and Figure 1b shows the details of the neural network architecture.

(a) Glyph-based visualization of the velocity field at time 0.

(b) Forward FTLE scalar field computed over 1,000 cycles.

FIG. 2: Visualizations of the Double Gyre data set showing the two counter-rotating gyres (Figure 2a) and the Lagrangian coherent structures as approximated by the ridge of the finite-time Lyapunov exponent (FTLE) scalar field (Figure 2b).

TABLE 1: Network training and computational performance results. We present the number of seeds (#Seeds), file cycle interval (Interval), number of training samples (#Samples), the training time (Train), trained model storage space (Model), and the inference performance (Inference) details of our experiments. The training time is measured for 100 epochs and increases linearly with the number of training samples. Importantly, our method costs 10.5MB memory for storing the trained model regardless of the number of training samples, potentially significantly reducing the storage space for large-scale time-varying vector fields. The inference time for 2,000 new particle trajectories interpolated across 1,000 cycles is presented. The interpolation of each location along a particle trajectory advances the particle by the length of the file cycle interval.

#Seeds	Interval	#Samples (M)	Train (hrs)	Inference (s)	Model (MB)
5,000	30	1.65	0.44	0.54	10.5
10,000	30	3.30	0.86	0.54	10.5
10,000	50	2.00	0.55	0.38	10.5

(a) $Lagrangian_{long}$ tests.

(b) $Lagrangian_{short}$ tests.

FIG. 3: Visualization of the errors mapped to the particle trajectory start location for three sampling strategies applied to generate both training and testing data sets. Figures 3a and 3b, show results for the $Lagrangian_{long}$ and the $Lagrangian_{short}$ flow map extraction strategies, respectively. The columns (left to right) represent $uniform$, $random$, and $sobol$ sampling for training seeds. The rows (top to bottom) represent $uniform$, $random$, and $sobol$ sampling for testing seeds. For example, in Figure 3a, column 1 row 3 shows the result of using uniform seeding for training sample generation and sobol seeding for testing reconstruction when using the $Lagrangian_{long}$ strategy. Each figure shows the spatial domain $[0, 2] \times [0, 1]$. The testing data contains 2,000 seeds for $random$ and $sobol$, and uses a $[50 \times 40]$ grid for $uniform$. The error is measured by aggregated along the trajectories (Equation 4) and is encoded in the visualization using the color and area of each circle mark. Overall, we find the $sobol$ or the Sobol quasirandom sequence strategy performs the best as a training and testing data sampling strategy across both flow map extraction approaches. However, we find the studied strategies can result in poor extrapolation for particles placed on the boundary.

FIG. 4: Loss versus epoch plots considering multiple learning rates for the two flow map extraction strategies. We use the learning rates 10^{-3}, 10^{-4}, and 10^{-5}. The training data set is generated by placing 5,000 seeds using the *sobol* method and file cycle interval is set to 30.

(a) 5,000 seeds.

(b) 10,000 seeds.

FIG. 5: The error plot evaluated various combinations of the learning rate and the batch size for $Lagrangian_{long}$ and $Lagrangian_{short}$ approaches. The errors are evaluated over 2,000 seeds and aggregated along the trajectories ((Equation 4)). The labels on the y-axis use 10^X format to show the error. We use the format N_B_LR to label each set of tests, where N is the number of seeds, B is the batch size, and LR is the learning rate. The top 1% of errors in each experiment are treated as outliers and have been removed for analysis. A batch size of 200 with the learning rates of 10^{-3} and 10^{-4} are optimal for training data sets with 5,000 seeds and 10,000 seeds, respectively, using the $Lagrangian_{long}$ approach. A batch size of 300 with the learning rate 10^{-4} is optimal for the $Lagrangian_{short}$ approach.

(a) Particle trajectory reconstruction error mapped to particle start location when varying the number of seeds used to generate training data.

(b) FTLE scalar field derived using trajectories inferred from the model.

FIG. 6: Visualization of particle trajectory reconstruction error mapped to particle start locations (6a) and the corresponding FTLE scalar fields derived from trajectories inferred by the model (6b), when varying the number of seeds used to generate training data. Each figure shows the spatial domain $[0, 2] \times [0, 1]$. The models are trained with a file cycle interval of 30 and the best combination of hyperparameter settings identified in Section 4.2. We evaluate reconstruction error using 2,000 seeds visualized as circle marks in 6a. The color and radius of the circles encode the error aggregated along the trajectories (Equation 4). The top 1% of errors are treated as outliers and have been removed for analysis from each experiment. The FTLE is calculated by placing a uniform grid with size $[256 \times 128]$. The model's performance is related to the flow behavior in the domain, and reconstruction errors are higher in regions with greater separation, notably for the $Lagrangian_{short}$, which suffers from error propagation.

FIG. 7: Violin plots of inference error evaluated for models trained using data generated with varying the number of seeds. The errors are calculated along the trajectories using Equation 4. The labels on the y-axis use 10^X format to show the error. The error is shown as a distribution using violin plots with the minimum, maximum, and median errors. The evaluation is performed using 2,000 random test seeds. The top 1% of errors are treated as outliers and have been removed for analysis from each experiment. Our results indicate the inference accuracy can improve from increasing the number of seeds used to train the model.

FIG. 8: Visualization of inferred trajectories and the ground truth for the Double Gyre with different numbers of seeds used to train the model. The seeds were randomly placed using Sobol seeding strategy. The colors of model inferred trajectories indicate the distance between the model inferred end location and the ground truth. In nearly all cases, our trained models can reconstruct trajectories almost visually identical to the ground truth.

(a) Resulting error maps when varying the file cycle intervals used to generate training data.

(b) FTLE scalar field derived using trajectories inferred from the model.

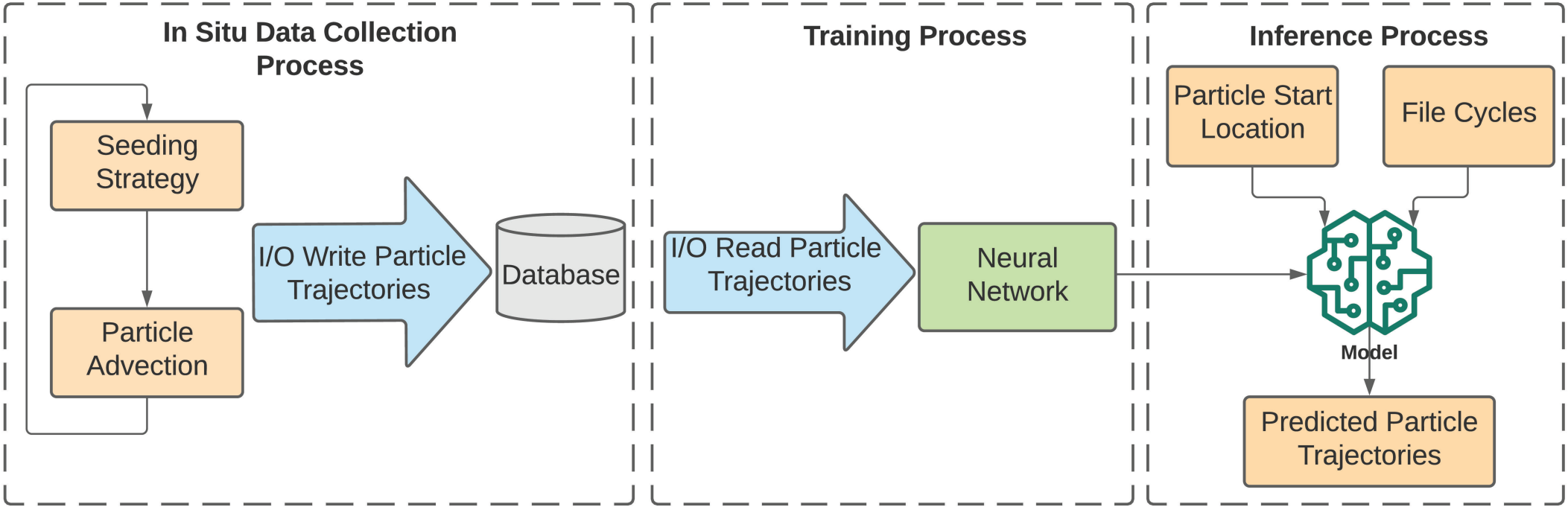
FIG. 9: Visualization of particle trajectory reconstruction error mapped to particle start locations (9a) and the corresponding FTLE scalar fields derived from trajectories inferred by the model (9b), when varying the file cycle interval used to generate training data. Each figure shows the spatial domain $[0, 2] \times [0, 1]$. The models are trained using 10,000 seeds and the best combination of hyperparameter settings identified in Section 4.2. We evaluate reconstruction error using 2,000 seeds visualized as circle marks in 9a. The color and radius of the circles encode the error aggregated along the trajectories (Equation 4). The top 1% of errors are treated as outliers and have been removed for analysis from each experiment. The FTLE is calculated by placing a uniform grid with size $[256 \times 128]$. The model's performance is related to the flow behavior in the domain, and reconstruction errors are higher in regions with greater separation. Notably, $Lagrangian_{short}$ tests with a short interval suffer from error propagation and accumulation.

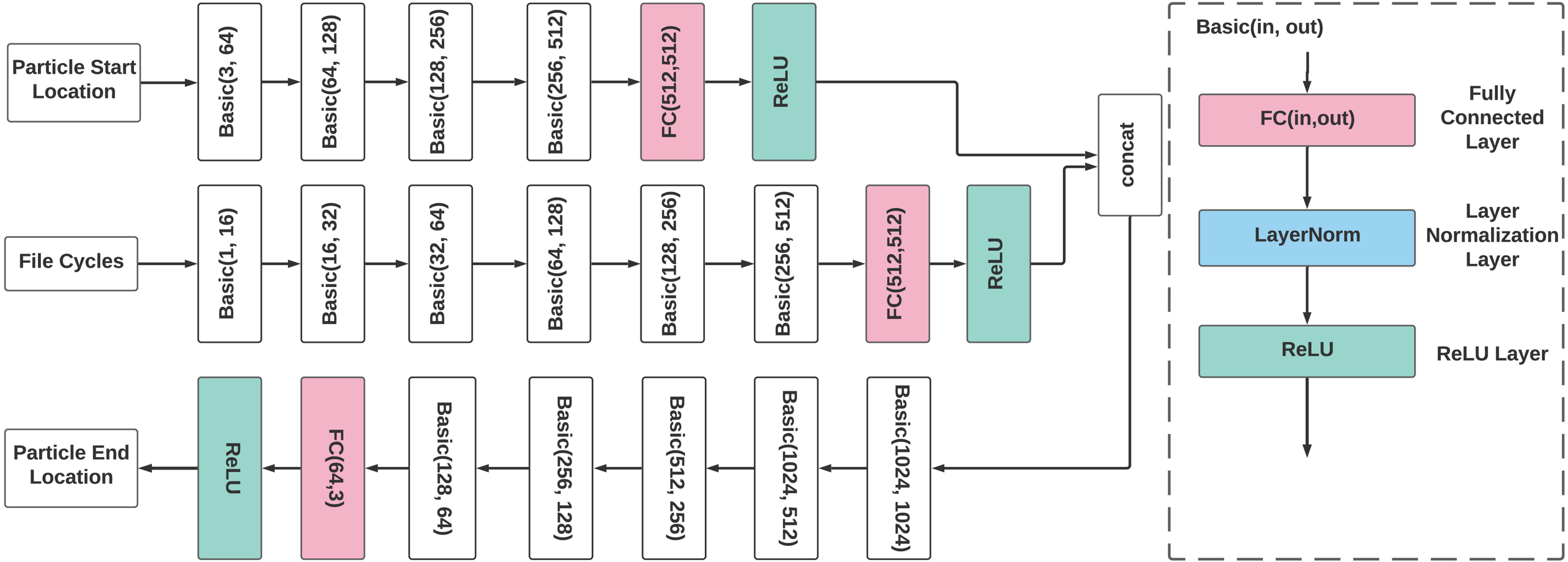
FIG. 10: Violin plots of inference error evaluated for models trained using data generated for varying file cycle intervals. The errors are calculated along the trajectories using Equation 4. The labels on the y-axis use 10^X format to show the error. The error is shown as a distribution using violin plots with the minimum, maximum, and median errors. The evaluation is performed using 2,000 random test seeds. The top 1% of errors are treated as outliers and have been removed for analysis from each experiment. Although the accuracy of $Lagrangian_{long}$ does not varying significantly with the considered file cycle intervals for the Double Gyre, the global error of the model trained using $Lagrangian_{short}$ decreases in accuracy as the length of the file cycle interval increases, but the local error increases with longer integration durations between file cycles.

FIG. 11: The average reconstruction error over file cycles for the Double Gyre data set with varying file cycle intervals. The errors are calculated by averaging distances between the model generated end locations and the ground truth at each file cycle. Evaluations are performed over 2,000 test seeds. For the $Lagrangian_{long}$ approach, errors do not propagate over file cycles. Results of different file cycle intervals have a similar trend. In contrast, errors are propagated in the $Lagrangian_{short}$ approach, and shorter file cycle interval results in more significant errors over time.

FIG. 12: Visualization of inferred trajectories and the ground truth for the Double Gyre with different file cycle intervals. The seeds were randomly placed using the Sobol seeding strategy. The colors of model inferred trajectories indicate the distance between the model inferred end location and the ground truth. Our trained model can reconstruct trajectories almost visually identical to the ground truth.

FIG. 13: Visualization of inferred trajectories and the ground truth for the ensemble member #200 vector field.


In Situ Data Collection Process
Seeding Strategy
Particle Advection
I/O Write Particle Trajectories
Database
Training Process
I/O Read Particle Trajectories
Neural Network
Inference Process
Particle Start Location
File Cycles
Model
Predicted Particle Trajectories

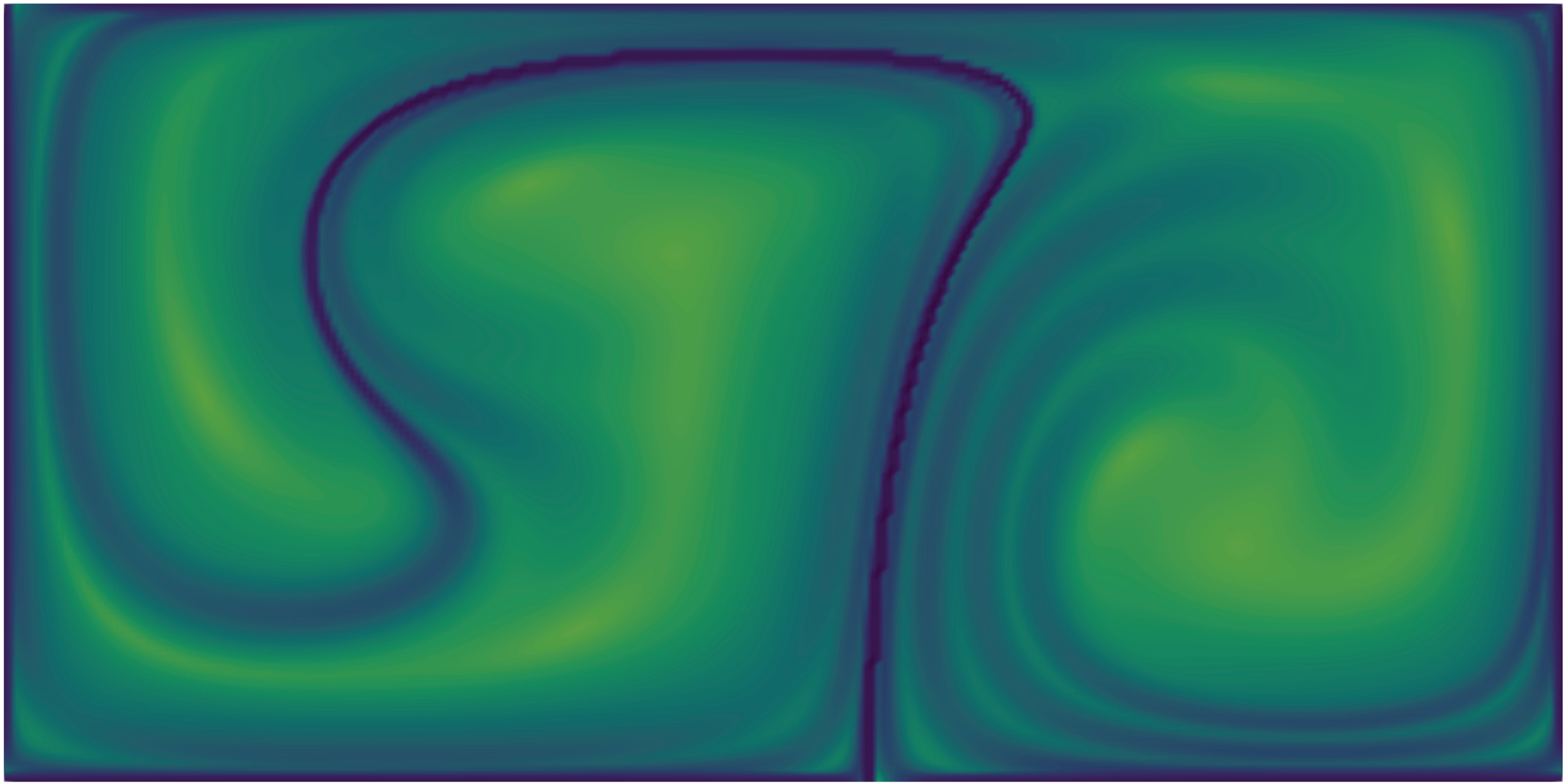

Particle Start Location
Basic(3, 64)
Basic(64, 128)
Basic(128, 256)
Basic(256, 512)
FC(512,512)
ReLU
File Cycles
Basic(1, 16)
Basic(16, 32)
Basic(32, 64)
Basic(64, 128)
Basic(128, 256)
Basic(256, 512)
FC(512,512)
ReLU
concat
Particle End Location
ReLU
FC(64,3)
Basic(128, 64)
Basic(256, 128)
Basic(512, 256)
Basic(1024, 512)
Basic(1024, 1024)
Basic(in, out)
FC(in,out)
Fully Connected Layer
LayerNorm
Layer Normalization Layer
ReLU
ReLU Layer

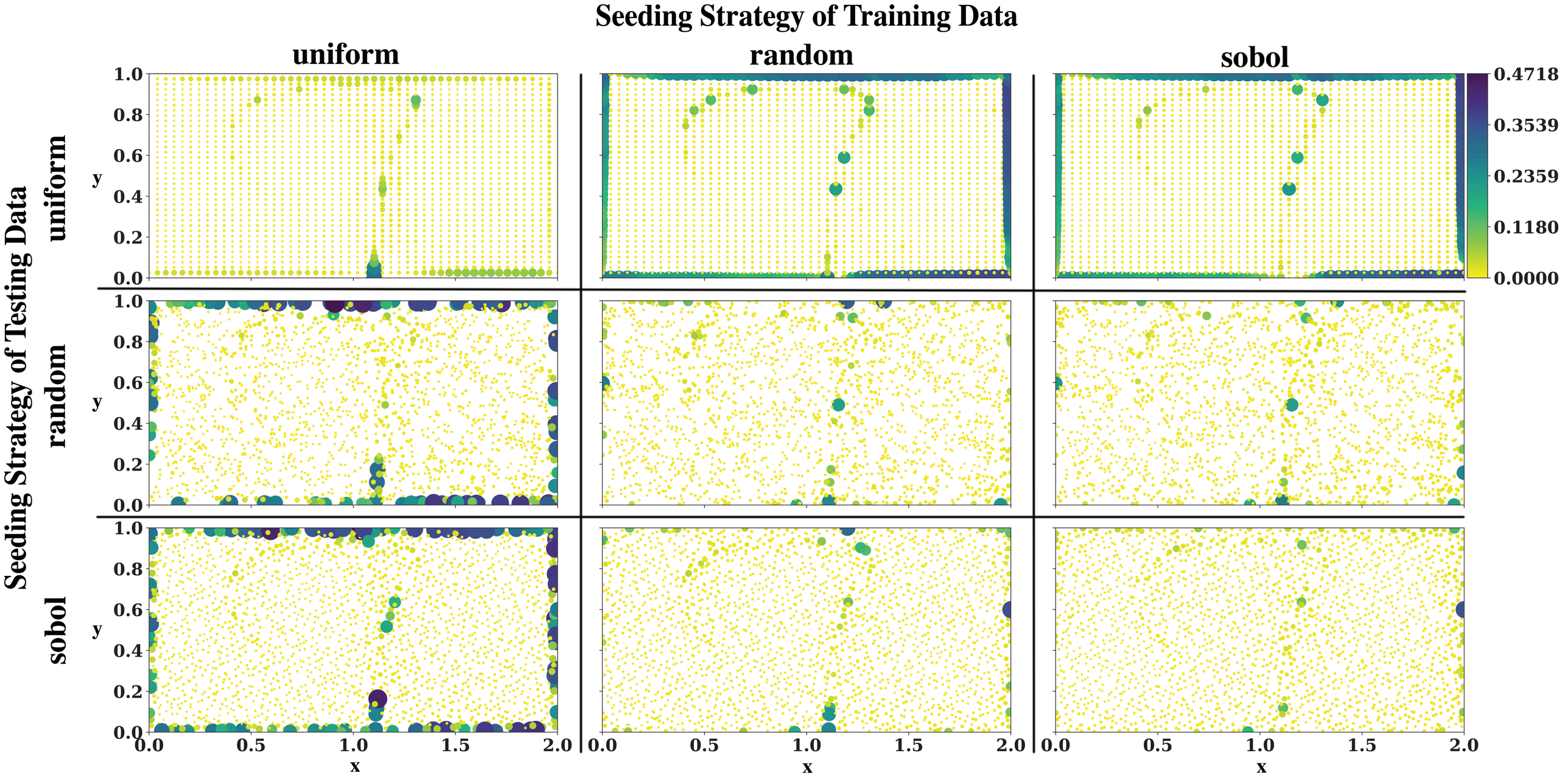
Velocity Magnitude

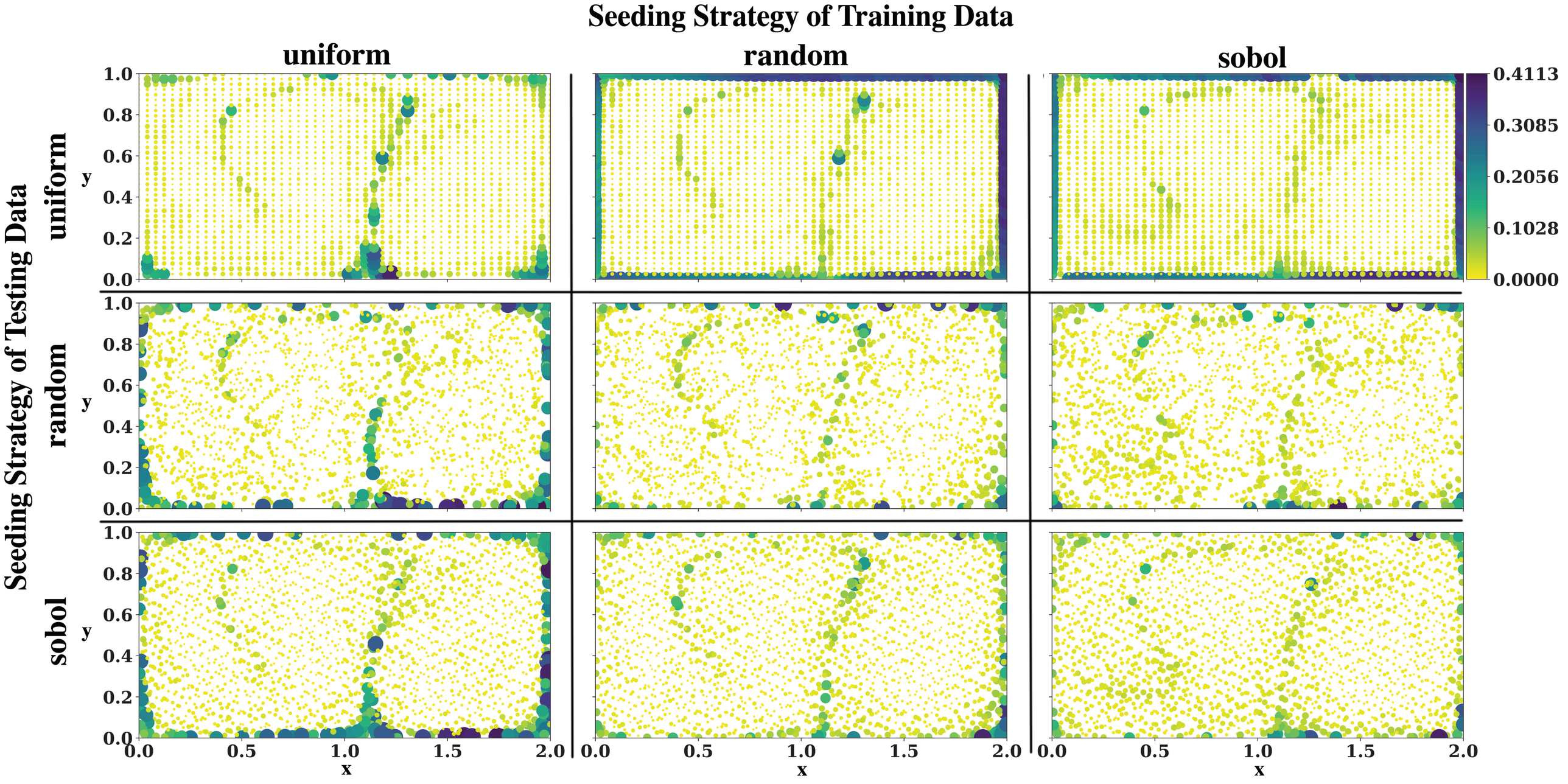
0.0 0.05 0.10 0.15 0.20 0.25 0.31

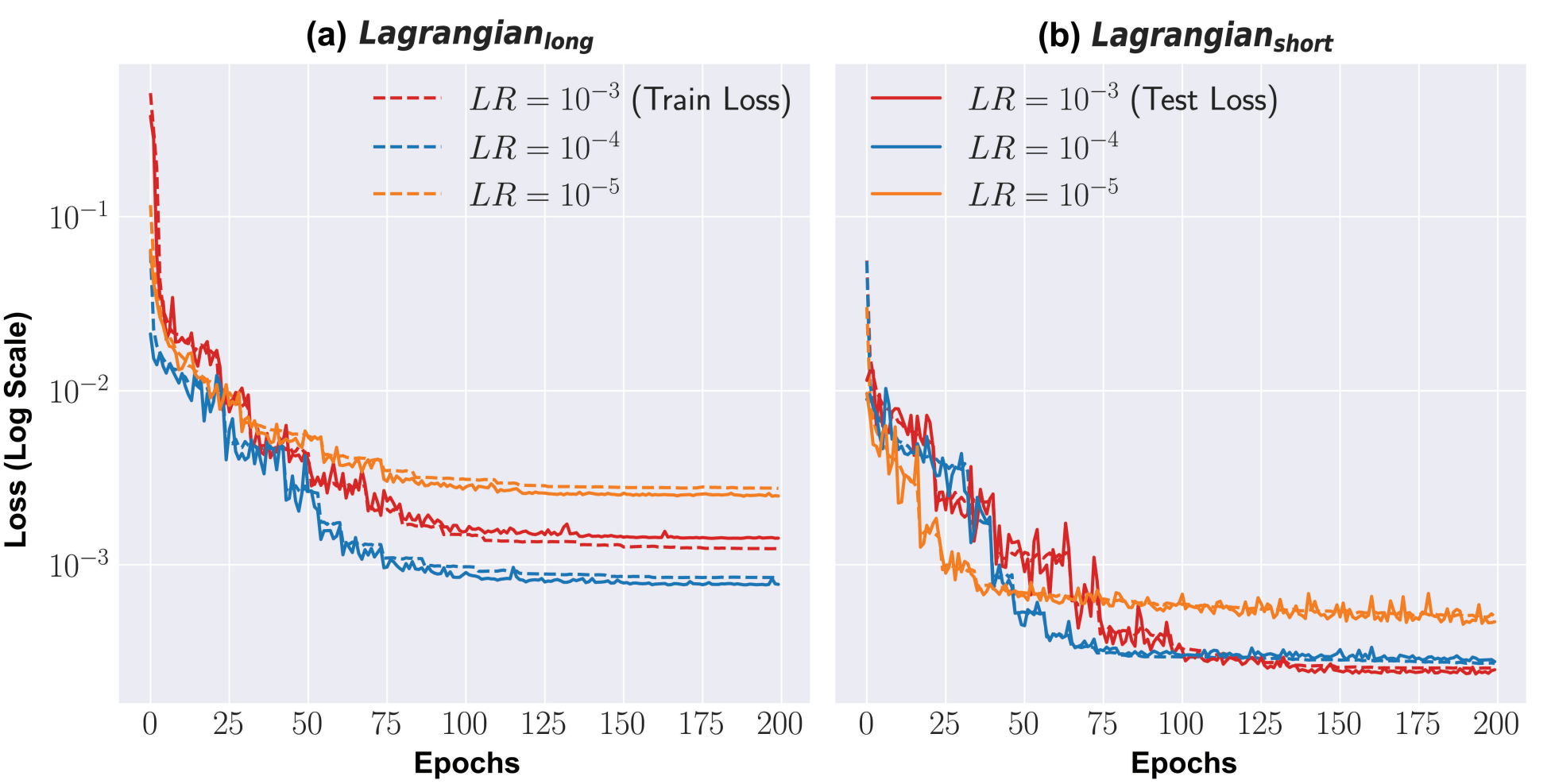
FTLE

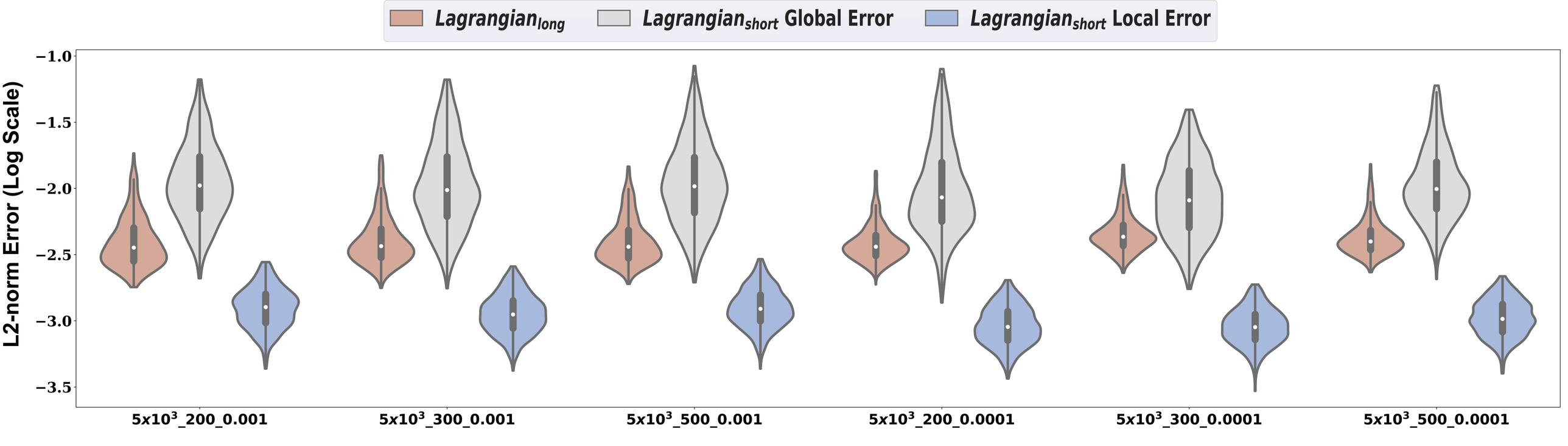
-0.13 0.0 0.1 0.2 0.3 0.4 0.53

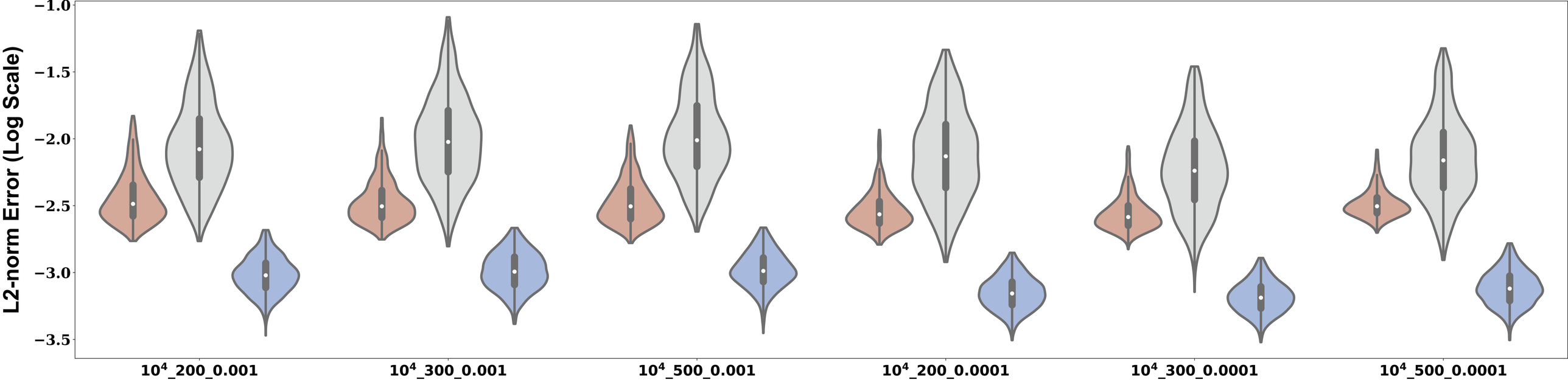


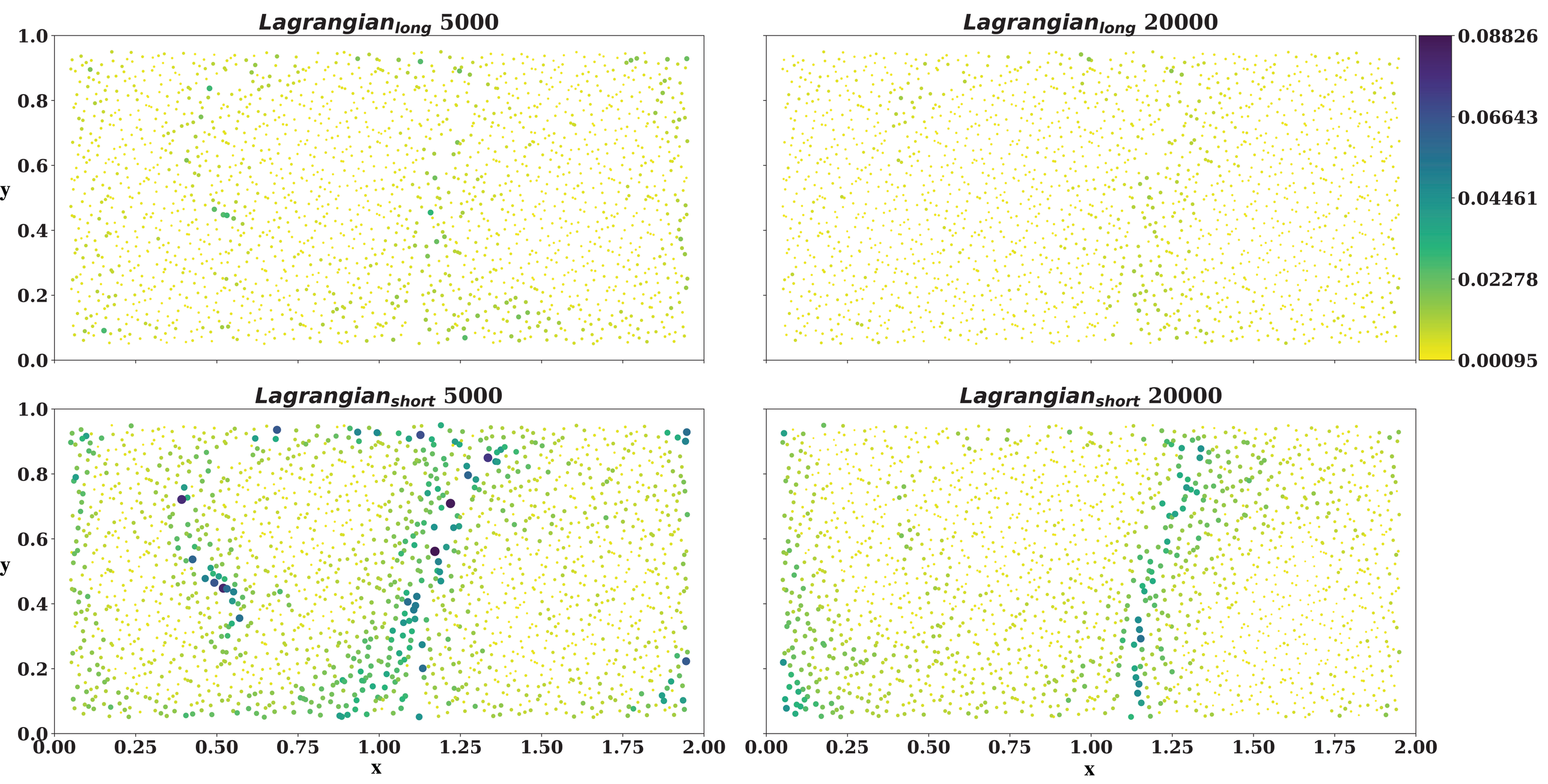

Seeding Strategy of Training Data
uniform
random
sobol
Seeding Strategy of Testing Data
uniform
random
sobol
y
x
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
0.4718
0.3539
0.2359
0.1180
0.0000

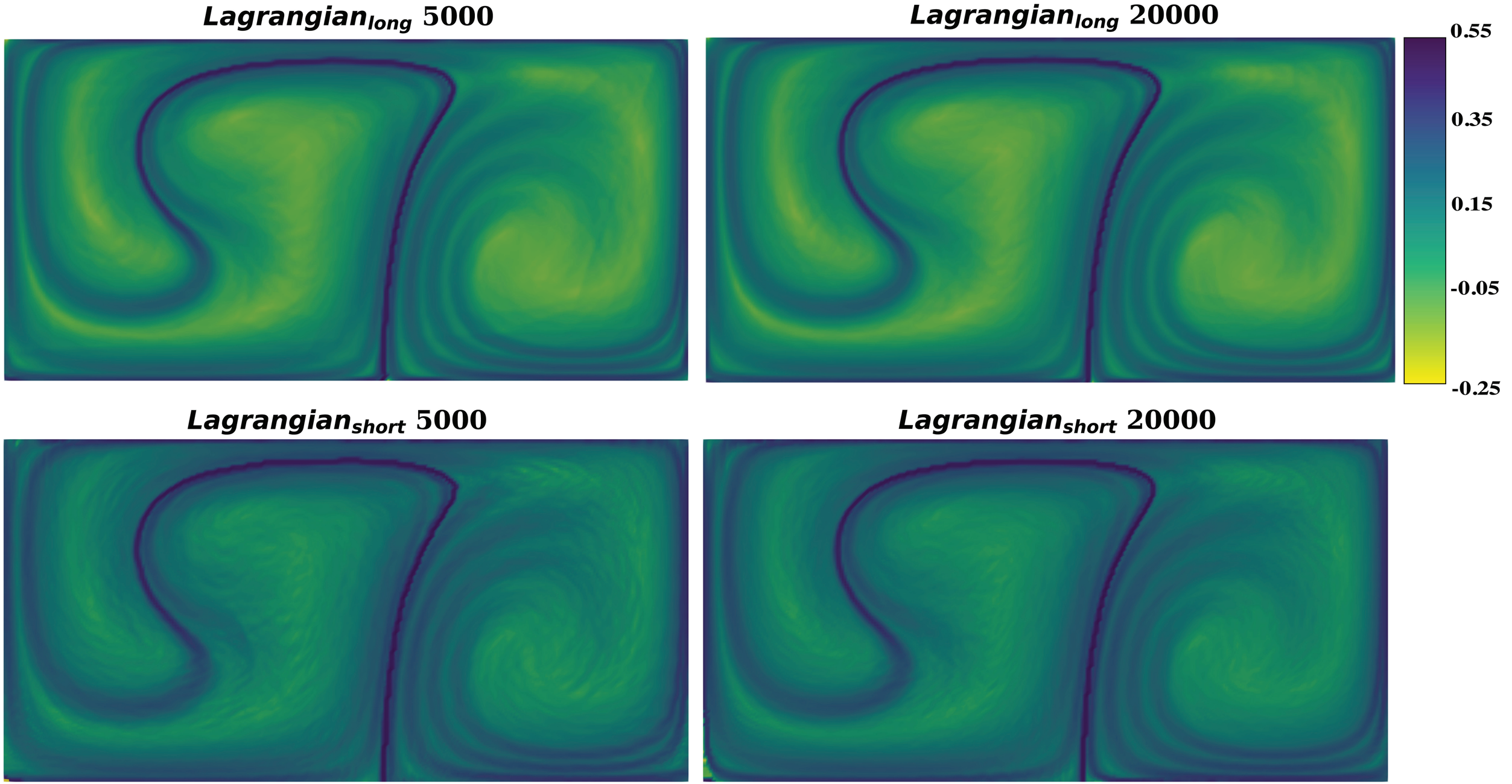

Seeding Strategy of Training Data
uniform
random
sobol
Seeding Strategy of Testing Data
uniform
random
sobol
y
x
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
0.4113
0.3085
0.2056
0.1028
0.0000

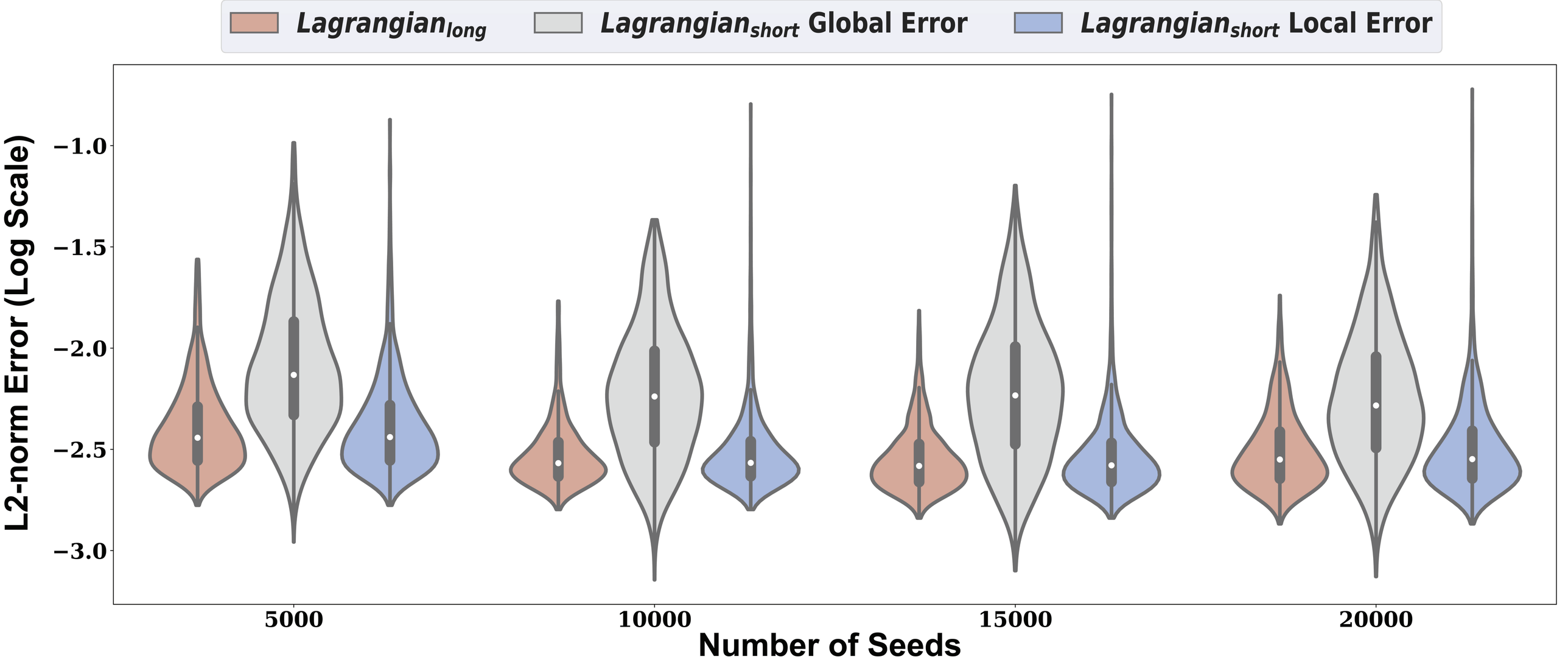

(a) Lagrangian_long
(b) Lagrangian_short
$LR = 10^{-3}$ (Train Loss)
$LR = 10^{-4}$
$LR = 10^{-5}$
$LR = 10^{-3}$ (Test Loss)
$LR = 10^{-4}$
$LR = 10^{-5}$
Loss (Log Scale)
10^{-1}
10^{-2}
10^{-3}
0
25
50
75
100
125
150
175
200
Epochs
Epochs


Lagrangian_long
Lagrangian_short Global Error
Lagrangian_short Local Error
L2-norm Error (Log Scale)
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
5x10^3_200_0.001
5x10^3_300_0.001
5x10^3_500_0.001
5x10^3_200_0.0001
5x10^3_300_0.0001
5x10^3_500_0.0001


L2-norm Error (Log Scale)
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
10⁴_200_0.001
10⁴_300_0.001
10⁴_500_0.001
10⁴_200_0.0001
10⁴_300_0.0001
10⁴_500_0.0001


Lagrangian_long 5000
Lagrangian_long 20000
Lagrangian_short 5000
Lagrangian_short 20000
y
x
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.08826
0.06643
0.04461
0.02278
0.00095


Lagrangian_long 5000
Lagrangian_long 20000
0.55
0.35
0.15
-0.05
-0.25
Lagrangian_short 5000
Lagrangian_short 20000


Lagrangian_long
Lagrangian_short Global Error
Lagrangian_short Local Error
L2-norm Error (Log Scale)
−1.0
−1.5
−2.0
−2.5
−3.0
5000
10000
15000
20000
Number of Seeds

Ground Truth
Seeds
Model Inferred

$Lagrangian_{long}$ 5000

$Lagrangian_{long}$ 20000

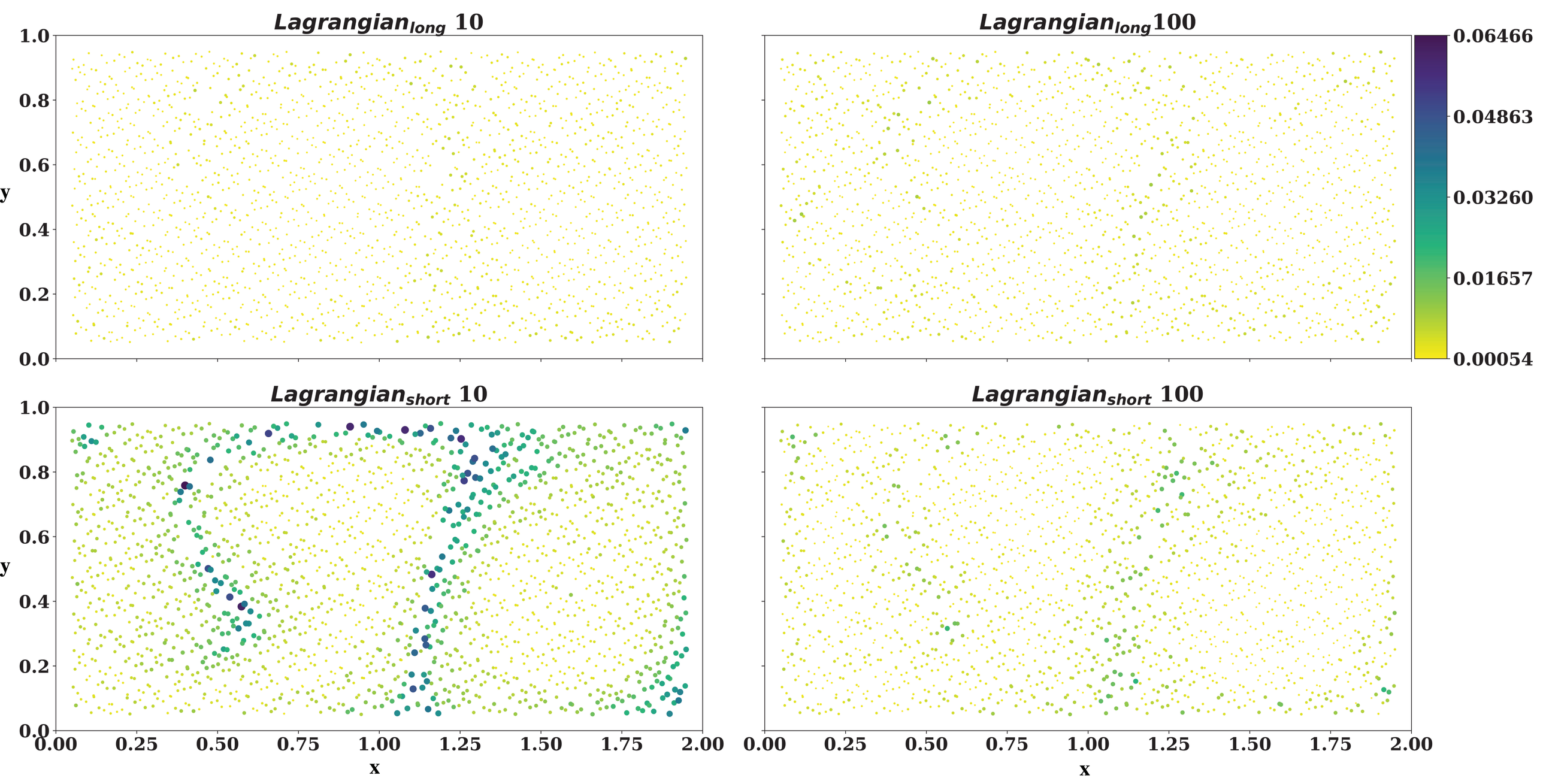
$Lagrangian_{short}$ 5000

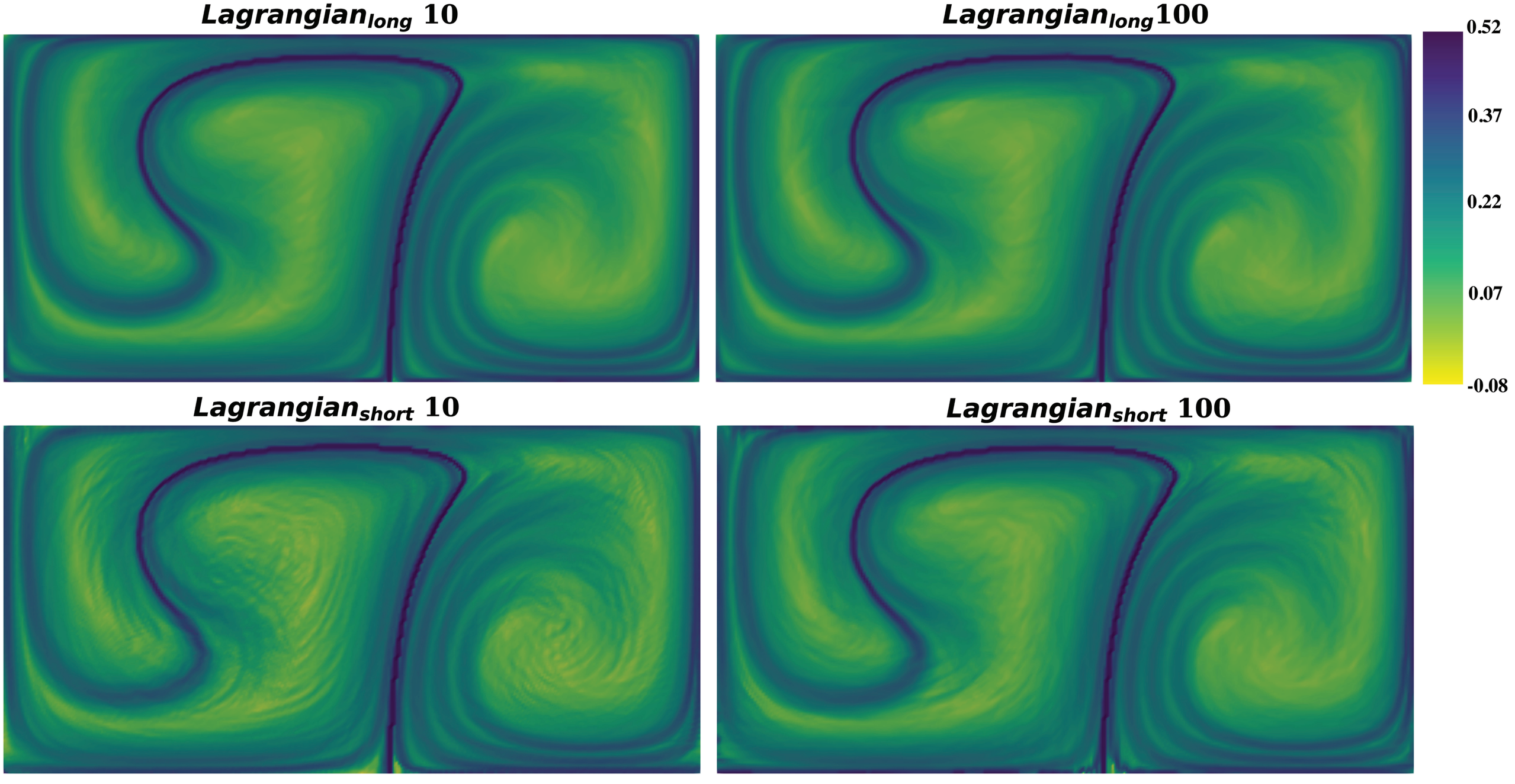
$Lagrangian_{short}$ 20000

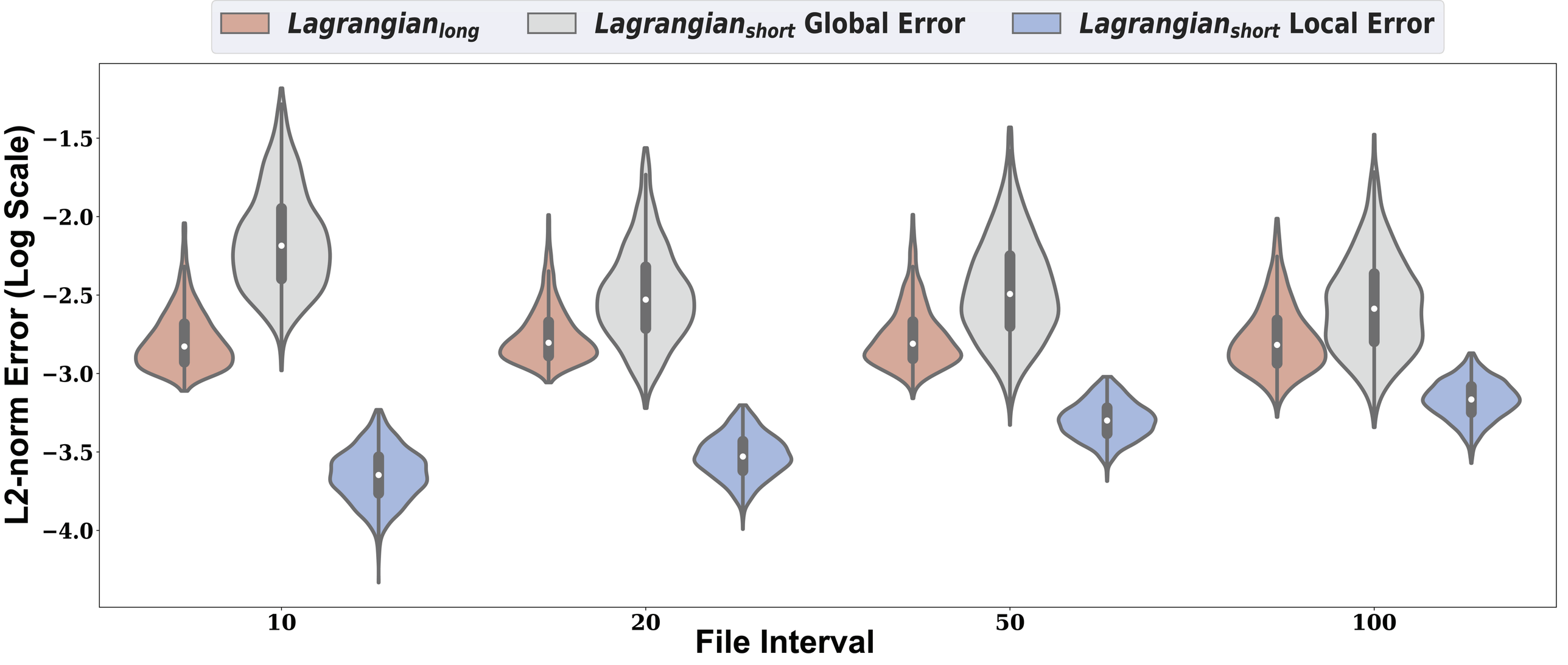
1.0
0.8
0.6
0.4
0.2
0.0

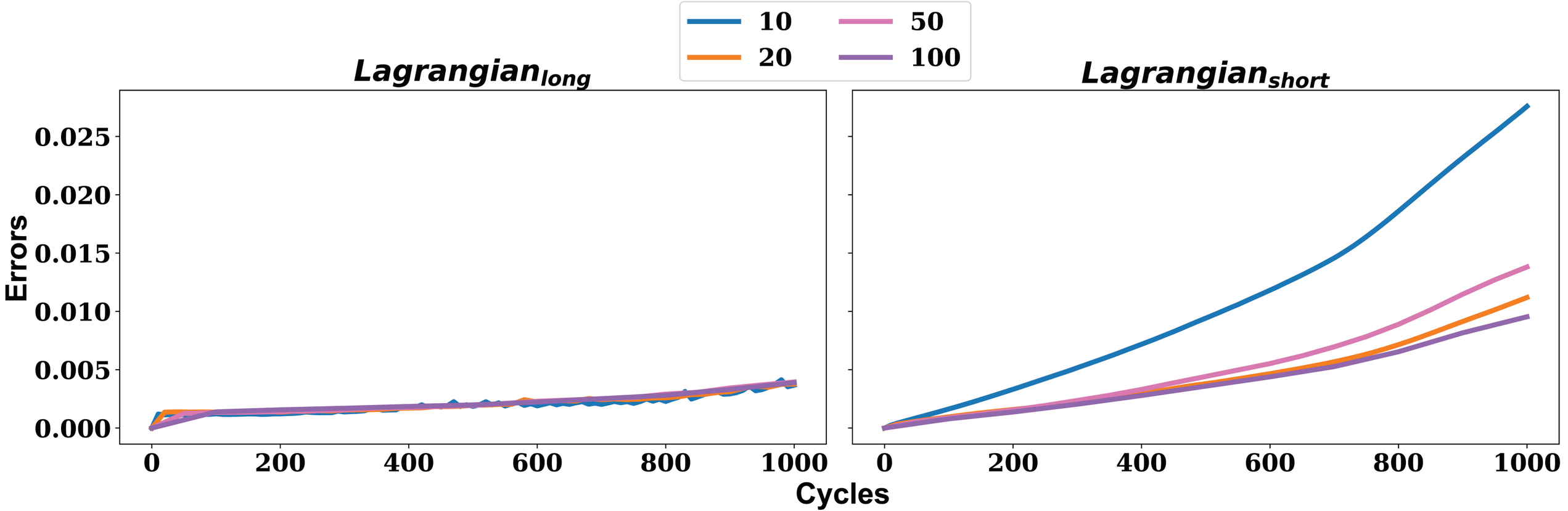
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00


Lagrangian_long 10
Lagrangian_long100
Lagrangian_short 10
Lagrangian_short 100
0.06466
0.04863
0.03260
0.01657
0.00054
y
x
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Lagrangian$_{long}$ 10
Lagrangian$_{long}$100
Lagrangian$_{short}$ 10
Lagrangian$_{short}$ 100
0.52
0.37
0.22
0.07
-0.08


Lagrangian_long
Lagrangian_short Global Error
Lagrangian_short Local Error
L2-norm Error (Log Scale)
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
10
20
50
100
File Interval


10
20
50
100
Lagrangian_long
Lagrangian_short
Errors
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
Cycles

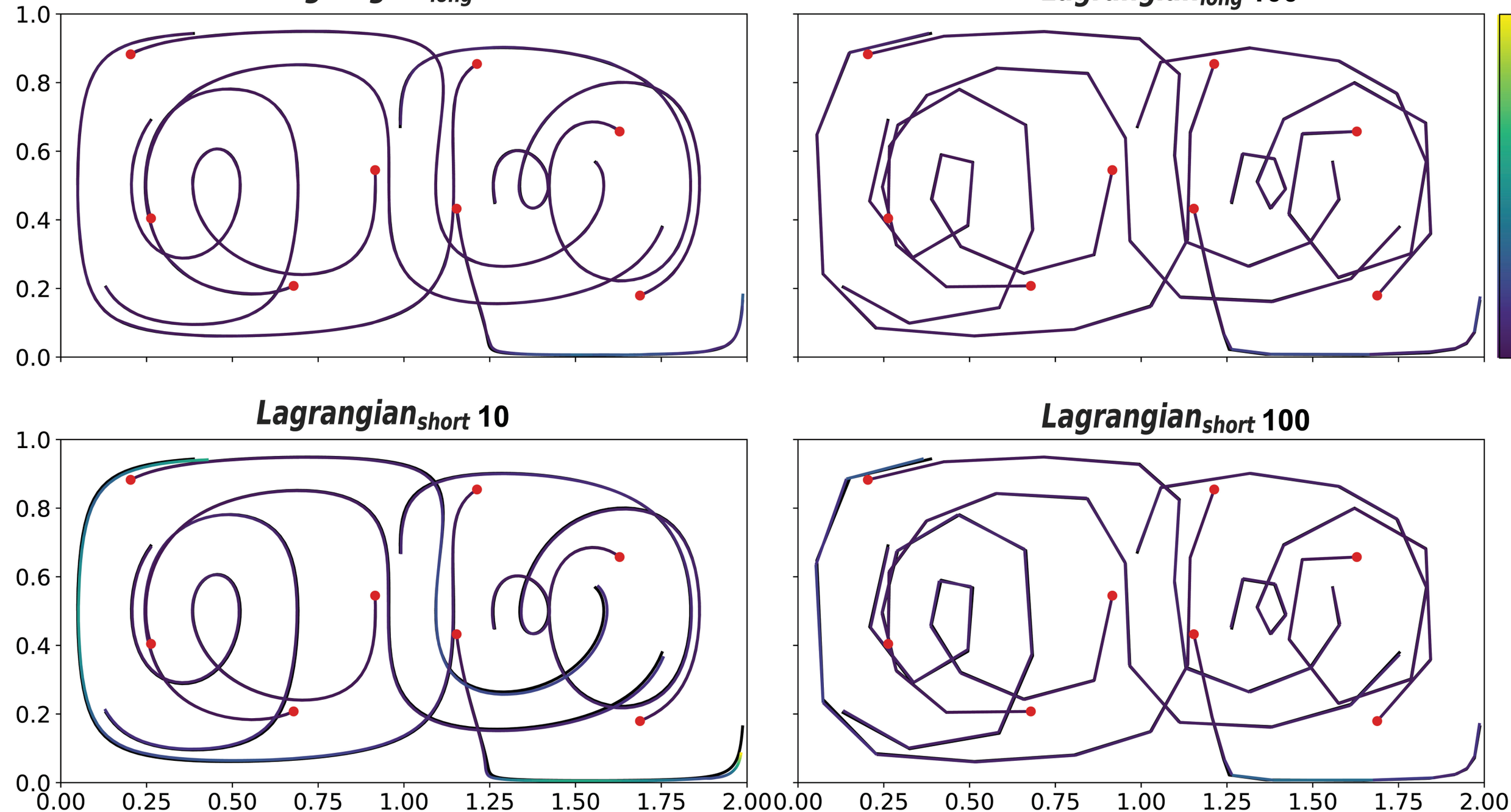
Ground Truth
Seeds
Model Inferred

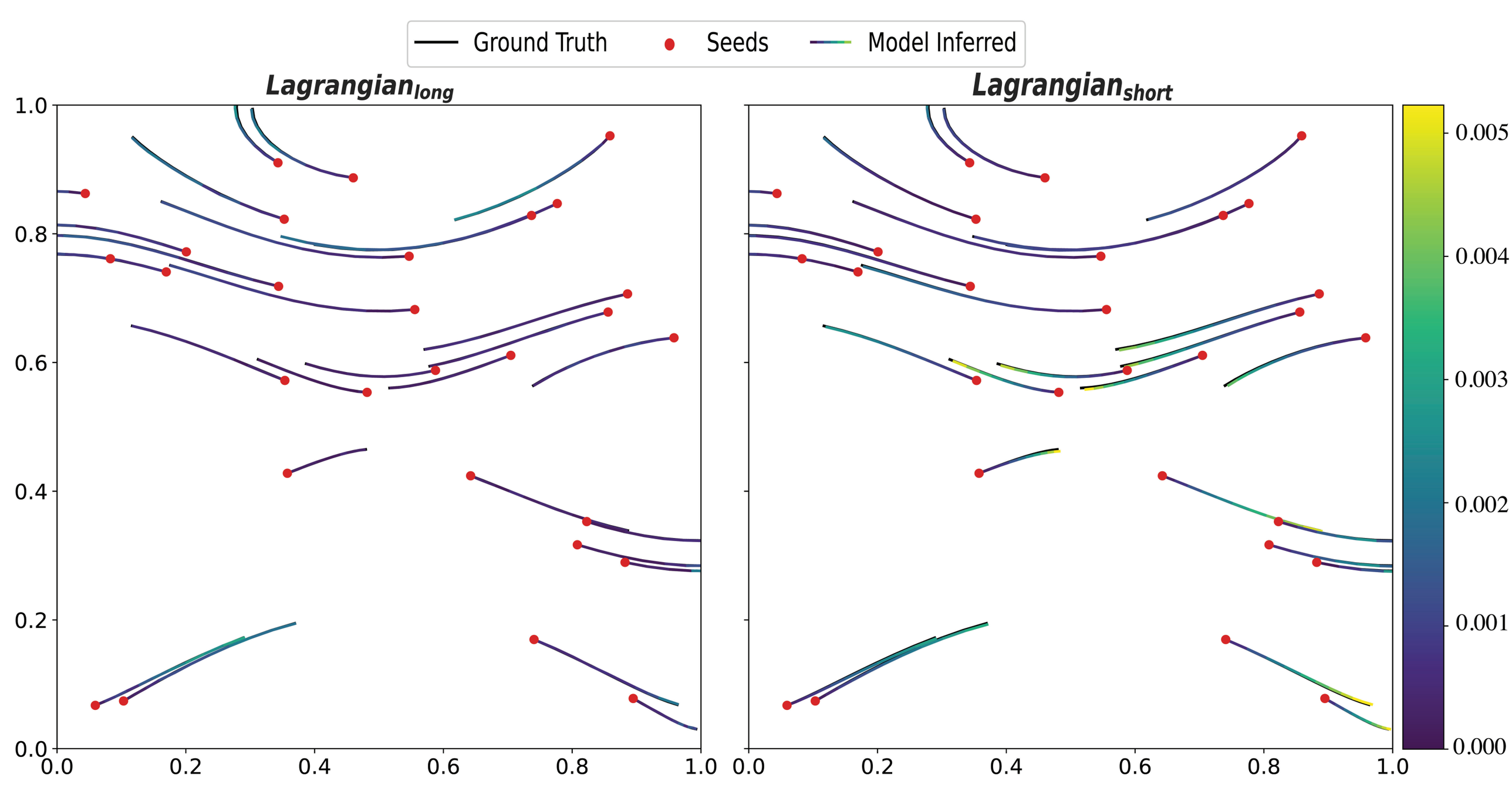
Lagrangian$_{long}$ 10

Lagrangian$_{long}$ 100

Lagrangian$_{short}$ 10

Lagrangian$_{short}$ 100




Ground Truth
Seeds
Model Inferred
Lagrangian$_{long}$
Lagrangian$_{short}$
0.0
0.2
0.4
0.6
0.8
1.0
0.000
0.001
0.002
0.003
0.004
0.005