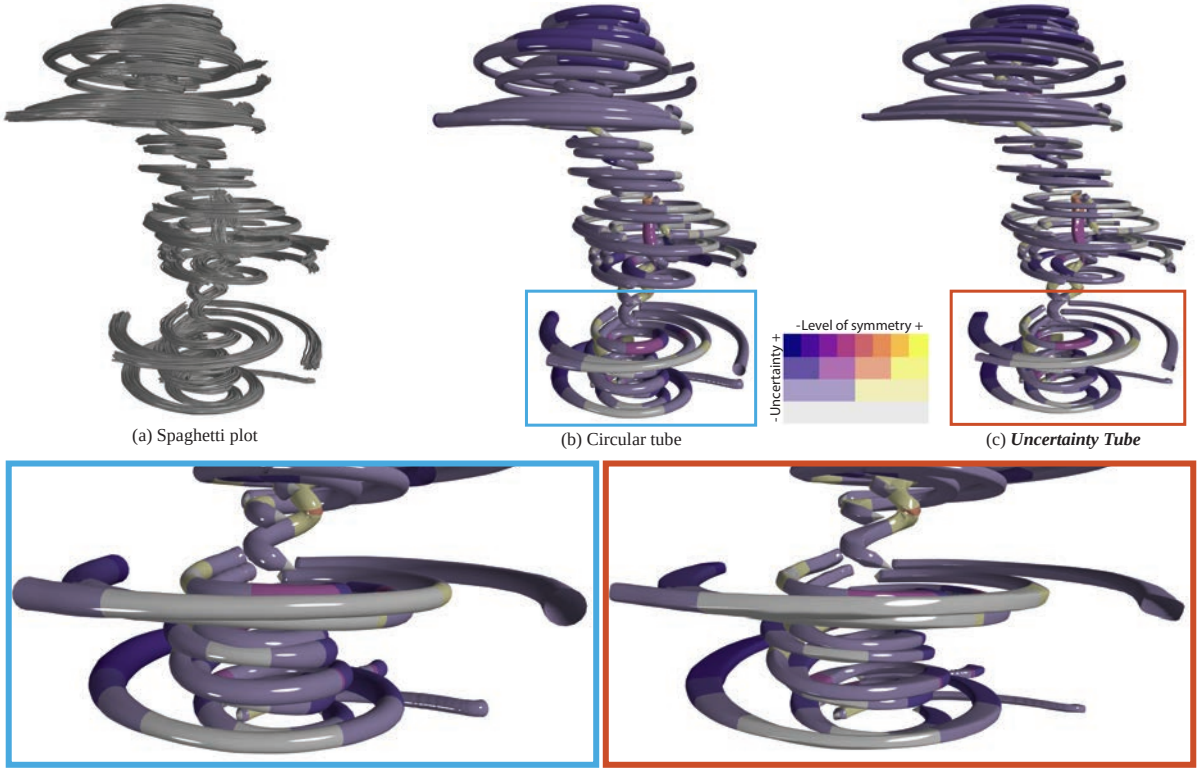
Uncertainty Tube Visualization of Particle Trajectories

Jixian Li* SCI Institute
Timbwaoga Aime Judicael Ouermi† SCI Institute
Mengjiao Han‡ Argonne National Laboratory
Chris R. Johnson§ SCI Institute



(a) Spaghetti plot (b) Circular tube (c) ***Uncertainty Tube***

Figure 1: Comparison of flow map uncertainty visualization techniques for the **tornado** dataset. This figure compares (a) a spaghetti plot of ensemble members, (b) a circular tube, and (c) our *uncertainty tube* for visualizing model uncertainty. Previous methods face challenges such as visual clutter (a) or the assumption of symmetric uncertainty (a, b), but our *uncertainty tube* (c), constructed using superellipses, provides a more accurate visualization of asymmetric uncertainty. Its superelliptical shape distinctly improves the visualization of the uncertainty orientation and its evolution along trajectories, as highlighted in the boxes. The visualization is further enhanced with a color palette that uses gray for low uncertainty, blue for large asymmetric uncertainty, and yellow for large symmetric uncertainty.

Abstract

Predicting particle trajectories with neural networks (NNs) has substantially enhanced many scientific and engineering domains. However, effectively quantifying and visualizing the inherent uncertainty in predictions remains challenging. Without an understanding of the uncertainty, the reliability of NN models in applications where trustworthiness is paramount is significantly compromised. This paper introduces the *uncertainty tube*, a novel, computationally efficient visualization method designed to represent this uncertainty in NN-derived particle paths. Our key innovation is the design and implementation of a superelliptical tube that accurately captures and intuitively conveys nonsymmetric uncertainty. By integrating well-established uncertainty quantification techniques, such as Deep Ensembles, Monte Carlo Dropout (MC Dropout), and Stochastic Weight Averaging-Gaussian (SWAG), we demonstrate the practical utility of the *uncertainty tube*, showcasing its application on both synthetic and simulation datasets.

Index Terms: Uncertainty visualization, vector field data, machine learning.

1 Introduction

Understanding and analyzing flow field data is fundamental for numerous scientific and engineering disciplines, including fluid dynamics, atmospheric science, and material processing. Traditional computational fluid dynamics (CFD) simulations are often computationally intensive, a limitation that has led researchers to explore more efficient paradigms. This exploration has given rise to neural networks (NNs) as a transformative tool in this domain, driven by their capacity to overcome these computational bottlenecks. NNs are now widely employed for tasks such as learning and emulating

*e-mail: jixianli@sci.utah.edu
†e-mail: touermi@sci.utah.edu
‡e-mail: hanm@anl.gov
§e-mail: crj@sci.utah.edu

complex turbulent fluid dynamics, enabling the rapid reconstruction of intricate flow fields from limited data, performing super-resolution on coarse simulation outputs, and serving as highly efficient surrogate models that can predict flow behavior orders of magnitude faster than conventional methods [38, 47, 64, 8]. Notably, recent work, such as Han et al. [26, 27], leverages NNs to learn Lagrangian-based flow maps, enabling efficient and robust particle tracing in time-varying fields. These data-driven models demonstrate remarkable accuracy and speed, making them increasingly indispensable for accelerating discovery and design cycles in fluid dynamics.

Despite these advancements, a significant challenge remains in providing a comprehensive understanding of the confidence associated with NN predictions in flow fields. Although NNs can effectively capture complex dynamics, their outputs are typically deterministic and lack explicit representations of predictive uncertainty. To address this limitation, several uncertainty quantification (UQ) methods have been developed for NNs, aiming to provide measures of prediction reliability [1, 18, 50, 17, 41, 35]. These techniques aim to estimate the confidence a model has in its output, distinguishing between inherent data variability and the model's uncertainty due to limited knowledge. However, effectively communicating these complex, often multidimensional uncertainty estimates to researchers, particularly for dynamic elements such as particle trajectories, remains a key challenge.

A recent work in uncertainty-aware deep neural representation of vector fields by Kumar et al. [34], which is closely related to our problem, utilizes circular tubes with varying radii to represent the variation in steamlines. While the circular tube provides valuable insights into trajectory variability, it assumes symmetric uncertainty bounds and can oversimplify and misrepresent scenarios where uncertainty is inherently asymmetric.

To address this limitation, this paper introduces the *uncertainty tube*: a novel, computationally efficient visualization method designed to represent prediction uncertainty in (NN)-derived particle paths. We focus our use case on neural network-based trajectories, but our method is generalizable to ensembles from other sources, such as simulations. This *uncertainty tube* is designed to accurately capture nonsymmetric uncertainty distributions, highlighting the direction of variations. The contribution of the paper includes:

1. The uncertain estimation from ensemble trajectories and the design of the *uncertainty tube* using superellipses and color palettes for accurate and enhanced visualization.

2. The use of three datasets (**synth, tornado, half cylinder**) to compare the uncertainty quantification methods (Monte Carlo dropout, Deep Ensembles, and SWAG) and demonstrate the capability of the *uncertainty tube* to accurately convey uncertainty in particle trajectories.

2 Related Works

2.1 Flow Field Visualization

2.1.1 Lagrangian Flow Reconstruction and Visualization

Eulerian and Lagrangian reference frames are commonly used to represent time-varying flow fields. In the Eulerian representation, velocity fields are stored, and particle paths are computed by integrating over time. In contrast, the Lagrangian representation uses flow maps to store particle start and end positions over a given time interval, enabling trajectory computation through interpolation. Each approach has its advantages and limitations. The Eulerian method is computationally efficient but requires a dense temporal resolution to achieve accurate trajectory reconstruction [11, 49, 2, 58, 54, 61]. The Lagrangian method offers a good accuracy–storage trade-off for exploring temporally sparse datasets [2, 52, 61, 60] and directly supports feature extraction [16, 62, 22, 15, 31]. As a result, it has received increasing attention in recent years.

In Lagrangian representation, the flow maps are typically computed in situ, while particle trajectories are reconstructed post hoc through interpolation. The accurate and fast reconstruction of new trajectories from the flow maps is an important component of post hoc analysis. Several methods have been proposed to enhance reconstruction accuracy and accelerate neighbor lookup in particle trajectory reconstruction, including multiresolution refinement through interpolation [3], parametric curve representations [7], and efficient neighborhood search using k-d trees [9]. However, these methods still face challenges in supporting real-time interpolation and visualization of flows. To address these challenges, scientists have been exploring deep-learning-based approaches [27, 26], which will be described in the next subsection.

2.1.2 Deep Learning for Flow Visualization

In recent years, deep learning has gained significant traction in the field of flow visualization [39]. They have been applied to a wide range of tasks, such as optimizing data access patterns to improve performance in distributed memory particle advection [30] and segmenting streamlines [37]. Deep learning has also been leveraged for selecting representative sets of particle trajectories [59], often using clustering approaches informed by learned features [23, 36]. In addition, many of the deep-learning techniques have been extended to directly integrate physical and conservation laws into the NN [51, 4, 57]. These physics-informed deep-learning approaches allow the models to learn from data while respecting underlying physical principles, leading to improved robustness and accuracy. Given the scale of flow datasets, data reduction and reconstruction have become prominent areas of focus. Several studies have demonstrated the use of low-resolution vector fields [21, 19, 29] or 3D streamlines [24, 55] to reconstruct high-resolution flow fields. In addition, recent work has explored temporal super-resolution of time-varying vector fields [25, 5]. Jakob et al. [31], for example, upsampled 2D FTLE scalar fields derived from Lagrangian flow maps by employing efficient super-resolution architectures. More recently, Sahoo et al. [56] introduced a method for compressing and reconstructing time-varying flow fields using implicit neural representations, demonstrating the promise of NNs for scalable and accurate flow reconstruction.

Effectively visualizing flow map data depends on two key factors: (1) accurately reconstructing particle trajectories and (2) enabling interactive visualization and exploration of these trajectories, as discussed in the previous section. Han et al. [27] were the first to employ a multilayer perceptron (MLP) architecture to reconstruct Lagrangian-based flow maps for a 2D analytical dataset. Their follow-up work [26] extended this by validating the approach on diverse 2D and 3D datasets and introducing a web-based viewer for interactive flow visualization using fast neural inference. While deep learning has been increasingly applied to flow visualization, existing works rarely explore model uncertainty in detail. In this work, we build upon the model proposed by Han et al. [26] and extend it with a specific focus on evaluating model uncertainty.

2.2 Estimating Uncertainty in Machine Learning Models

Addressing the uncertainty in deep learning is an active research area, with various approaches offering different trade-offs in rigor, cost, and performance. In this paper, we focus on the model's epistemic uncertainty instead of aleatoric uncertainty resulting from noise and randomness in the data. Here, we list some of the common uncertainty estimation methods:

Deep Ensembles [35]: This method involves training multiple NNs independently from different random initializations. The variance of predictions across these ensemble members quanti-

fies uncertainty. Deep Ensembles are empirically robust and well-calibrated, but incur high computational costs during training due to the requirement of multiple full models.

Monte Carlo Dropout (MC Dropout) [17]: MC Dropout adapts dropout regularization for uncertainty estimation during inference. This approach is computationally efficient because it does not require additional training if dropout is already being used. Despite these benefits and its wide adoption, others argue that its predictive distribution does not align with a true Bayesian posterior [14]. Still, MC Dropout remains a valuable technique for capturing model uncertainty in many applications.

Bayesian Neural Networks (BNNs) and Approximations: BNNs aim to model a probability distribution over network weights, directly accounting for epistemic uncertainty. However, the exact Bayesian inference in NNs is currently intractable due to the high dimensionality and complexity of the posterior distribution. To address this limitation, various approximation methods such as Variational Inference [6], Markov Chain Monte Carlo [65], Laplace Approximation [40], and Stochastic Weight Averaging-Gaussian (SWAG) [41].

In this paper, we utilized the two methods used in Kumar et al. [34], the Deep Ensembles and MC Dropout. We also added SWAG to demonstrate how our visualization can be applied to different ensemble-based methods. Both MC Dropout and SWAG introduce minimal computational overhead, which allows interactive exploration and visualization of flow map uncertainty.

2.3 Uncertainty Visualization

Effectively visualizing uncertainty is important for data analysis and decision-making [12, 53]. Researchers have developed numerous visualization techniques to communicate uncertainty, ranging from fundamental statistical glyphs to more advanced probabilistic representations [28]. Notable contributions in the field include the early review by Pang et al. [46], discussions on challenges and approaches by Johnson and Sanderson [32], and the comprehensive taxonomy of uncertainty visualization approaches by Potter et al. [48]. More recently, Kamal et al. [33] provided a survey on recent advances and ongoing challenges in the field, highlighting the continuous importance of depicting data quality and variability to ensure accurate interpretation.

Vector and flow field uncertainty visualization is challenging due to the directional and dynamic nature of these fields. These challenges are further exacerbated when considering time-varying, multiple computational fields, and large-scale data. To convey uncertainty, most techniques focus on representing variability of both direction and magnitude. Early work explored glyph-based approaches, such as those by Wittenbrink et al. [66], for visualizing uncertainty in vector fields. More recently, Ouermi et al. [45] have advanced glyph-based uncertainty visualization for time-varying vector fields. To represent topological properties of uncertain fields, Otto et al. [43] and Otto et al. [44] introduced methods for uncertain 2D and 3D vector field topology. Mirzargar et al. [42] proposed curve boxplots as a generalization of traditional boxplots for ensembles of curves, providing statistical summaries of trajectory bundles. In addition to the depth-based boxplot idea, Ferstl et al. [13] introduced variability plots to cluster and characterize the major trends in the ensemble. Kumar et al. [34] have explored uncertainty-aware deep neural representations to aid in the visual analysis of vector field data. Extending Kumar et al.[34], which uses circular tubes with varying radii to represent uncertainty of integralines, we use the *uncertainty tube* to better represent the non-symmetric uncertainty.

3 Background

3.1 Deep-Learning-Based Largrangian Flow Maps

Han et al. [27] introduced the first multilayer perceptron (MLP)-based model to explore time-varying vector fields using Lagrangian-based flow maps, which they later improved for more accurate predictions and evaluated on multiple datasets [26]. In our paper, the uncertainty measurements are based on the flow map NN proposed by Han et al. [26], which is described below.

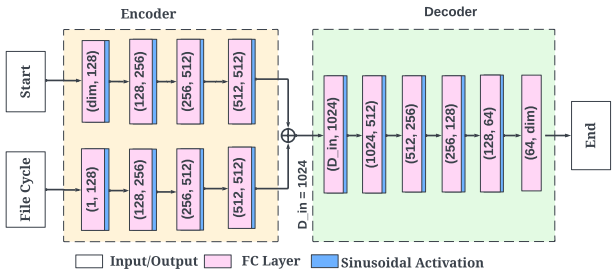


Figure 2: The MLP-based flow map NN proposed by Han et al. [26]. This image, used with the authors' permission, illustrates a network configuration with four encoding layers, six decoding layers, and a latent vector dimension of 1024. The architecture begins by taking two inputs: the particle's initial position (Start) and the number of file cycles (File Cycle). These inputs are first processed by the encoder, which transforms them into a latent vector denoted as *D_in*. This latent vector is then passed to the decoder, which outputs the particle's predicted position (End) at the queried file cycle. A sinusoidal activation function is applied after each fully connected (FC) layer, except for the output layer.

3.1.1 Training Data Generation

In our work, we adopt only the Lagrangian_long method introduced by Han et al. [26], which generates a single flow map by tracing long particle trajectories with uniform temporal sampling along each integral curve. For seeding, we use a Sobol quasirandom sequence, which, as shown in previous work [27], outperforms pseudo-random sequences and uniform grid sampling. Once the initial seeds are placed in the spatial domain, the particle trajectories are calculated by advancing them from time t to $t+\delta$, where δ represents one simulation time step (or cycle). Tracing starts from the initial time $t0$ to the final time T, with the results saved at each file cycle. The final training dataset is structured as an $m \times n$ array, where m is the number of seeds and n is the number of file cycles. Each training sample contains a start position s_i (where $0 \leq i \leq m-1$), the corresponding file cycle index c_j (where $0 \leq j \leq n-1$), and the end location ($\ell_{i,j}$). The training dataset is formatted as Eq. (1), allowing the model to learn both spatial and temporal patterns in the flow field.

$$\begin{aligned} Input = &\{\{s_0, c_0, \ell_{0,0}\}, \{s_0, c_1, \ell_{0,1}\}, ..., \\ &\{s_0, c_{n-1}, \ell_{0,n-1}\}, ..., \{s_{m-1}, c_{n-1}, \ell_{m-1,n-1}\}\}. \end{aligned} \tag{1}$$

3.1.2 Network Architecture

We adopt the MLP-based NN architecture proposed by Han et al. [26] (see Fig. 2). The encoder **E** takes as input the particle start locations and the corresponding file cycles, processing them through two distinct sequences of fully connected (FC) layers. The outputs of these sequences are concatenated to form a latent vector, which is then passed to the decoder **D**. The decoder predicts the particle end locations, which are compared to the ground truth using

the L1 loss. The model uses the sine activation function throughout. The network architecture is dynamic, featuring a configurable number of encoder and decoder layers, as well as a variable latent vector dimension.

3.2 Uncertainty Quantification for Deep Neural Networks

We employ three uncertainty quantification methods, Deep Ensembles, MC Dropout, and SWAG, to assess the uncertainty in the flow map prediction.

3.2.1 Deep Ensembles Method

A deep ensemble [35] consists of multiple, independently trained neural networks on the same dataset, each with different random initializations. This process leads each network to learn slightly varied representations of the data. During inference, an input is passed through every network, and the individual predictions are aggregated. The final prediction is the mean of these outputs, while the spread or disagreement among the predictions serves as a direct measure of the model's uncertainty. A high variance indicates low model confidence, whereas low variance suggests high confidence and model agreement.

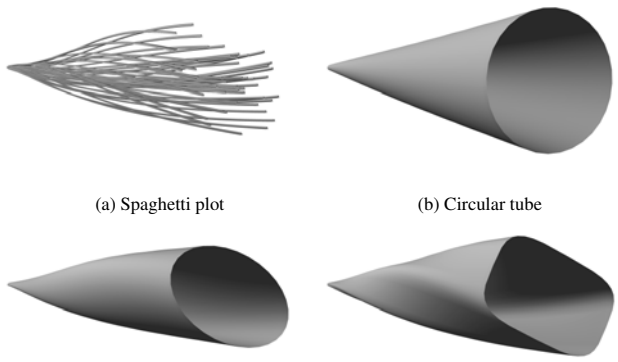
3.2.2 Monte Carlo Dropout

Monte Carlo Dropout (MC Dropout) [17] estimates the uncertainty in NNs by keeping dropout active during inference. For each forward pass a random set of neuron is deactivated, enabling the single network to behave like an ensemble of multiple sub-networks without the computational burden of training multiple full models. The MC Dropout predictive uncertainty is calculated by running multiple forward passes (Monte Carlo samples) for the same input and calculating the variation among the predictions.

3.2.3 Stochastic Weight Averaging-Gaussian

The Stochastic Weight Averaging-Gaussian (SWAG) [41] method is an innovative approach designed to quantify uncertainty in NNs while mitigating the high computational cost associated with traditional ensemble techniques. Unlike Deep Ensembles, which train multiple models independently, SWAG focuses on capturing a distribution over the NN's weights in a single training run. It achieves this by averaging the network's weights throughout training, particularly towards the end of the optimization process, and then fitting a multivariate Gaussian distribution to these weights. The Gaussian approximates the Bayesian posterior of the model weights. An ensemble of models can be sampled from the Gaussian without requiring the training of multiple models.

To leverage the SWAG method for uncertainty quantification, an NN is first trained as usual. After the initial training phase, the learning rate is often set to a high constant value, and the network continues training using stochastic gradient descent (SGD) for a few more epochs to explore the loss landscape more comprehensively, thereby avoiding being stuck in local minima. During this fine-tuning phase, snapshots of the network's weights are periodically saved and averaged to ensure stability and consistency. Once this process is complete, a covariance matrix is estimated from these collected weight samples, often using a low-rank approximation to ensure computational feasibility. For a new input, predictions are then generated by sampling multiple sets of weights from this approximated Gaussian distribution, effectively creating a "virtual ensemble" from a single trained model. The spread of these predictions, similar to a traditional ensemble, then indicates the model's predictive uncertainty.



(a) Spaghetti plot (b) Circular tube

(c) *Uncertainty tube* with $\tau = 2$ (d) *Uncertainty tube* with $\tau = 4$

Figure 3: For a given set of uncertainty samples (a), the circular tube (b) does not visually encode the twisty motion or asymmetric distribution. The elliptical tube (c) highlights the asymmetry with a small hint of twistiness. The superelliptical tube (d) highlights both asymmetry and twistiness.

4 Visualizing Flow Map Uncertainty

4.1 Uncertainties of the Predicted Trajectories

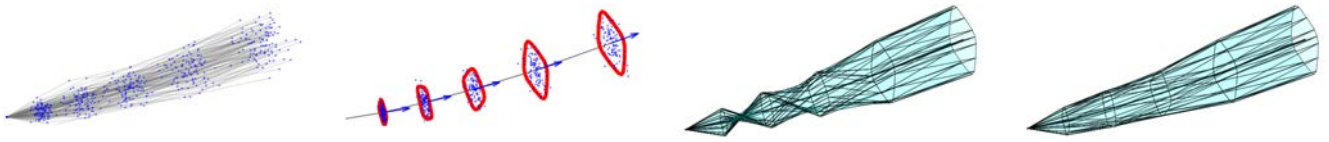
Kumar et al. [34] visualize the integral line uncertainty using a circular tube, such that the variation is encoded as the radius of the tube. Upon closer examination of the uncertainty samples from multiple uncertainty quantification methods, we noticed that most of the uncertainty obtained from these methods is not symmetric. Visualizing them as a round tube hides the asymmetric nature of the uncertainty distribution. We observed the asymmetry of the uncertainty across models, datasets, and UQ methods. For some datasets, we were able to utilize the asymmetry to enhance our understanding of the data and improve the training of our model. More examples of how we can use this information are presented in Sec. 6.

Figure 3a shows an example of nonsymmetrically distributed uncertainty. The gray uncertainty samples exhibit more uncertainty in one direction than the other. However, a spaghetti plot is not very effective here because of the complex occlusion patterns. Meanwhile, as Fig. 3b demonstrates, a circular tube is not very effective in representing asymmetric and twisty uncertainty, either. We need a visual encoding that better captures the asymmetry and twistiness while reducing the visual clutter introduced by visualizing all uncertainty samples. We also need to compute the statistical summary sufficiently fast to avoid compromising the efficiency of our NN flow map model. Therefore, we introduce the *uncertainty tube* as demonstrated in Fig. 3c and Fig. 3d.

4.2 Uncertainty Tube

The *uncertainty tubes* are constructed by building a superelliptical tube between two consecutive time steps, $t-\delta$ and t, starting from the seed location. For an ensemble of particle trajectories with N time steps, as illustrated in Fig. 4a, we start at the seed location $\mathbf{x}^{(0)}$, which is assumed to have no uncertainty. For each subsequent time $t = \{\delta, \cdots, N\delta\}$, We build the super elliptical tube between $t-\delta$ and t. Let $\mathbf{x}_i^{(t)}$ be uncertainty samples at t and $\bar{\mathbf{x}}^{(t)}$ be the mean of those points. We project all $\mathbf{x}_i^{(t)}$ to the plane orthogonal to the direction from $\bar{\mathbf{x}}^{(t-\delta)}$ to $\bar{\mathbf{x}}^{(t)}$ and passing through $\bar{\mathbf{x}}^{(t)}$. The projection is defined by

$$\mathbf{p}_i^{(t)} = \mathbf{x}_i^{(t)} - ((\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}^{(t)}) \cdot \mathbf{d})\mathbf{d}, \tag{2}$$



(a) Step 1: Raw sample trajectories (b) Step 2: Uncertainty superellipse (c) Step 3: Tube mesh before alignment (d) Step 4: *Uncertainty tube* mesh

Figure 4: *Uncertainty tube* construction steps. This figure illustrates the design process for constructing the *uncertainty tube*, starting from the raw trajectories shown in (a), followed by the projection to form superellipses in (b), and finally the alignment step in (c) and (d) to form the *uncertainty tube*.

where $\mathbf{d}$ is the unit normal vector to the plane. This projection is designed to capture the uncertainty in the orthogonal cross-section, but does not account for the variation along the normal direction ($\mathbf{d}$).

We calculate the covariance matrix of the projected points and its eigenvalue decomposition:

$$\mathbf{V}\Sigma\mathbf{V}^T = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})\cdot(\mathbf{p}_i - \bar{\mathbf{p}})^T, \tag{3}$$

where Σ and $\mathbf{V}$ are the eigenvalues and vectors, respectively. The right-hand side of Eq. (3) shows the covariance calculation with $\bar{\mathbf{p}}$ being the mean of the projected points. The result from the decomposition is then used to construct a superellipse according to

$$\mathbf{q}(\theta) = \mathbf{e}(\theta)\mathbf{V}^T + \bar{\mathbf{p}}, \tag{4}$$

where the function $e(\theta)$ is defined according to

$$\mathbf{e}(\theta) = \begin{pmatrix} 2\sigma_1|cos(\theta)|^{\frac{2}{\tau}}sgn(cos(\theta)) \\ 2\sigma_2|sin(\theta)|^{\frac{2}{\tau}}sgn(sin(\theta)) \end{pmatrix}, \theta \in [0, 2\pi], \tag{5}$$

where σ_1 and σ_2 are the diagonal entities of Σ. The function *sgn* returns the sign of its input value. The parameter $\tau \geq 2$ controls the shape of the superellipse; For greater values of τ the superellipse becomes more rectangular with sharp corners, whereas for $\tau = 2$ the shape reduces to the standard ellipse. The results in this paper use $\tau = 4$. The superellipse centered at the mean $\bar{\mathbf{p}}$ summarises the variance of the projected points. The superellipse representation provides a more accurate representation of the direction variation of the projected points compared to a standard circle. Additionally, its rectangular shape helps clarify orientation more effectively than a standard ellipse. These uncertainty superellipses are shown in Fig. 4b. To form the *uncertainty tube*, the superellipses at t and $t+\delta$ are sampled, and their corresponding boundary points are connected.

However, the sampled points $\mathbf{q}_j^{(t)}$ and $\mathbf{q}_j^{(t+\delta)}$ can be misaligned, leading to warped and twisted superelliptical tubes, as shown in Fig. 4c. To address this, we calculate an optimal circular shift and reverse orientation ordering that minimizes the alignment score according to

$$\begin{aligned} \{\hat{r}, \hat{s}\} &= \underset{r\in\{0,1\}, s\in\{0,1,\cdots,m-1\}}{\operatorname{argmin}} \sum_{j=1}^{m} \|\mathbf{q}_{\ell_j}^{(t+\delta,r)} - \mathbf{q}_j^{(t)}\|, \\ \ell_j &= (j+s) \bmod m. \end{aligned} \tag{6}$$

The subscript ℓ_j in Eq. (6) performs the circular shift at s, and a reverse ordering is employed if $r = 1$. The aligned *uncertainty tube* is shown in Fig. 4d.

Overall, the *uncertainty tube* significantly improves the representation of asymmetric uncertainty compared to the circular tube. Moreover, the superelliptical tube's rectangular design distinctly shows the uncertainty orientation and its evolution along the pathline more effectively than the circular and elliptical tubes, as demonstrated in Fig. 3.

Seeds/Steps	10	50	100	150	200
10	401	440	561	608	653
100	625	744	868	971	1124
300	936	1220	1667	1823	2122
500	1029	1509	1957	2454	2994

Table 1: *Uncertainty tube* computation costs in milliseconds (ms)

4.3 Computation Efficiency

The computation of the *uncertainty tube* introduces a one-time cost per user's query. The overhead mainly consists of two parts: 1. the *UQ time* measures the time it takes to obtain the uncertainty samples. 2. The *meshing time* is the time it takes to compute the mesh and texture coordinates.

The UQ time is reported in Sec. 5 for different methods. Here, we report the meshing time on AMD Ryzen Threadripper 3970X 32-Core Processor using 32-core parallelization in Tab. 1. Each row represents the number of seeds. Each column represents the number of steps per trajectory. To compute the *uncertainty tube*, we use 50 uncertainty samples per trajectory. The number of uncertainty samples within the range of 10 to 100 has a minimal impact on the meshing time.

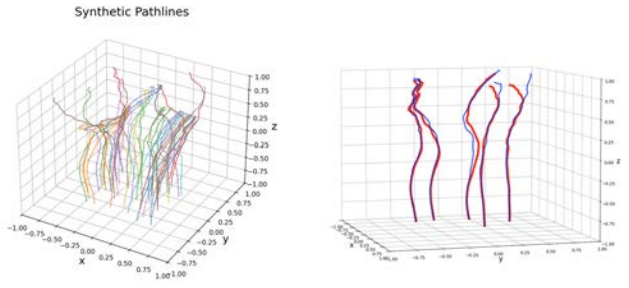
A typical setup in our data exploration stage uses 100 to 300 seeds to understand the global trend. All our datasets have 50 to 150 steps. We found that 30 to 50 uncertainty samples are sufficient to build a meaningful statistical model of the uncertainty. This means that *uncertainty tube* computation incurs a few seconds of overhead per query, which still fits within the interactive exploration scheme proposed in Han et al. [26] given ensemble samples from the UQ method.

We normally spend a much longer time interacting with the rendered scene than sending different queries. In our experiments, all *uncertainty tube* visualization renders at 120 frames per second on a 2023 MacBook Pro with an Apple M2 Max chip after the initial rendering pass.

4.4 Uncertainty Coloring

Sometimes the geometry alone is not sufficient for the analysis of uncertainty. For example, comparing Fig. 9a and Fig. 9c without color is challenging because the camera is positioned far away to reveal the dataset's global pattern, and the size difference is visually diminished. Therefore, we use color to help us compare uncertainties at a different granularity.

Inspired by value-suppressing uncertainty palettes (VSUP) [10], we employ a similar color map to represent the amount of uncertainty and the level of symmetry. Our primary task is to visualize the uncertainty of an ensemble of trajectories. Unlike the original VSUP, we suppress trajectories with low uncertainty; that is, if the prediction has low uncertainty, the colormap does not distinguish between the levels of symmetry. We chose a light gray color, indicating a low level of uncertainty. For high uncertainty, we use a linear color palette to determine the level of symmetry. The level of symmetry is determined by the ratio of the first two eigenvalues. 1



(a) 50 random pathlines

(b) Test data (red) compared to model prediction(blue)

Figure 5: Matplotlib demonstration of the **synth** dataset and training outcomes.

means symmetry, where the *uncertainty tube* would be round, and 0 means high asymmetry, indicating one major variation direction, resulting in a flat *uncertainty tube*. The uncertainty colormap first determines the color for the level of symmetry by linearly interpolating the color palette. Then the final color can be computed by interpolating between the color of symmetry and light gray according to the level of uncertainty. The level of uncertainty is the magnitude of the first eigenvalue, rescaled to a value between 0 and 1, according to the user-set threshold. For example, the user could map the 98th percentile of the data to 1 to reduce the impact of extreme values. In the visualization, using the viridis colormap as an example, gray represents low uncertainty, blue represents nonsymmetric uncertainty, and yellow represents symmetric uncertainty. Figure 6 demonstrates how our colormap is applied in a synthetic dataset.

5 Controlled Experiments

We first use a synthetic dataset, labeled as **synth**, to demonstrate different uncertainty quantification and visualization in a controlled setting. The **synth** dataset is a time-varying vector field in which the particles move in the positive z-direction. We introduce more complexity as the z value increases. Hence, we expect the trained model to exhibit more uncertainty in the positive z-direction. The domain is $[-1,1]$ for all axes. The seeding box is $[-0.5,0.5],[-0.5,0.5]$, and $[-1,-0.9]$ for the x, y, and z axes, respectively. Each pathline is traced 50 steps, including the seeds. Fig. 5a shows 50 random pathlines traced from the seeding box. For the training data, we use the Sobol method to generate $131,072$ seed locations. The testing dataset consists of $5,000$ uniformly sampled seeds inside the seeding box.

5.1 Deep Ensembles

To utilize the Deep Ensembles method, we independently trained 50 models on the same training dataset. To induce randomness in the training process, we randomly shuffle the order of the training dataset at each iteration. For each model, we use four encoder layers and four decoder layers, setting the latent dimension to 1024. We train a total of $10,000$ iterations. On two NVIDIA 3090 GPUs, training each model takes approximately 3.5 minutes. The total time for training 50 models is approximately 3 hours. Quantifying uncertainty using the Deep Ensembles has a significant computational time overhead. However, we also notice that the Deep Ensembles method reports uncertainty most faithful to our construct. We did not try to optimize the training parameters for the optimal training quality. On average, we expect an absolute difference of 0.027 between the predicted location and the truth. Figure 5b shows the quality of the trained model on five randomly selected testing pathlines. As expected, the error increases as z increases.

In practice, we often lack the testing data to assess the true error of the predictions. Thus, we rely on the uncertainty quantification to provide insight into the model's confidence. We sample 225 points within the seeding box and then evaluate the pathlines from all 50 models. We calculate the mean pathline across 50 models and use the 51 paths per seed to build the *uncertainty tube*. Figure 6a shows the *uncertainty tube* visualization, from which we observe that the models exhibit higher variation in the positive z direction (rightward), aligning with our expectation. We also noticed that in the high-uncertainty regions, the distribution of uncertainty trajectories is mostly asymmetric, as indicated by the predominance of blue hues toward the end. We selected one pathline with high uncertainty from the scene to demonstrate the bias in Fig. 7a.

In summary, the Deep Ensembles method provides a good demonstration of uncertainty, aligning with our data constructions, but at a relatively high computational cost, as noticed in other works that utilize the Deep Ensembles method.

5.2 MC Dropout

MC Dropout introduces the least amount of overhead among the three UQ methods used in this paper, provided that dropout is already part of the model during training. Evaluating 225 trajectories with 50 uncertainty samples takes less than 400 milliseconds. It requires modifying the model by appending dropout layers if dropout is not already part of the training.

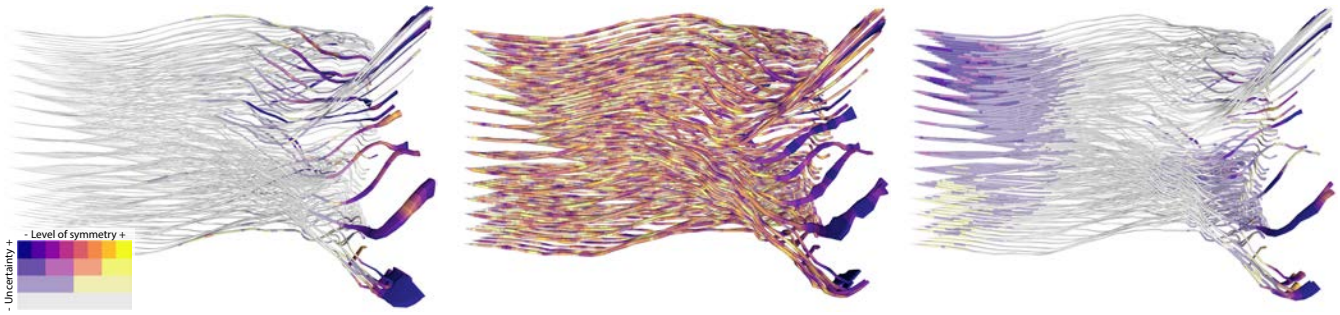
To conduct the MC Dropout experiments, we implemented two ways of appending dropout layers. The first way is to append a dropout layer after every activation. The second approach is to append a dropout layer after the last activation. Table 2 demonstrates the impact of adding dropout on the testing metric, namely, the absolute difference between the prediction and the truth. We confirm the finding in Kumar et al. [34] that adding dropout results in a decrease in prediction quality. For our model, dropping out after all activation layers at a rate of 0.001 has minimal impact on prediction quality, and it closely matches the original proposed MC Dropout method [17]. We use this configuration to produce the outcomes shown in Fig. 6b.

Figure 6b shows the pathlines evaluated from the same set of seeds used to demonstrate the Deep Ensembles method. The magnitude of uncertainty estimated by MC Dropout is larger than Deep Ensembles' result in the low-uncertainty region, as shown by the colorfulness of the *uncertainty tubes*. At the same time, the high-uncertainty region (bottom right) shows a smaller uncertainty magnitude compared to the result from Deep Ensembles. However, overall, the MC Dropout exhibits higher uncertainty at larger z values, as demonstrated by the change in size of the *uncertainty tubes*. Fig. 7b also shows that MC Dropout's result may be significantly different than the results of the other two methods.

In summary, MC Dropout is a straightforward method that incurs no additional training costs. Due to neural networks' efficient inference capability, the overhead of running multiple inference passes is minimal. However, while originally proposed as an approximate Bayesian method, it is difficult to rigorously argue that the samples from MC Dropout are drawn from a true Bayesian posterior distribution. This is because dropout randomly sets activations to zero, and the underlying theoretical approximations are often not strictly met in practice. Indeed, research by Folgoc et al. [14] has strongly argued that MC Dropout does not perform approximate Bayesian inference.

5.3 SWAG

Maddox et al. [41] introduced the SWAG method in 2019. Despite its simple and efficient utilization, we have yet to see it being used for visualizing uncertainty in the visualization community. This method introduces minimal overhead and does not require modification to the model. However, it requires careful tuning of hyperparameters. We report a hyperparameter study in Sec. 5.3.1.

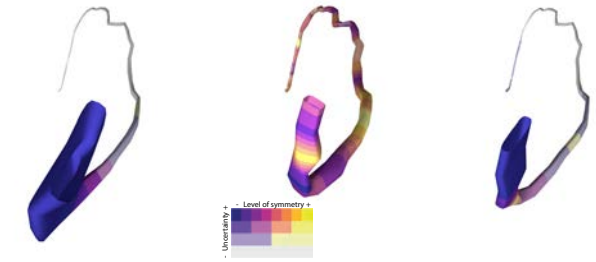


(a) Deep Ensembles (b) MC Dropout (c) SWAG

Figure 6: *Uncertainty tube* visualizations of 225 pathlines from **synth** dataset using three quantification methods. The z value, hence the introduced uncertainty, increases from left to right. Each tube is computed from 50 uncertainty samples.

dropout rates	0.1	0.05	0.04	0.03	0.02	0.01	0.005	0.001
all layers	0.034	0.031	0.030	0.029	0.029	0.028	0.028	0.027
last layer	0.032	0.028	0.029	0.029	0.028	0.028	0.028	0.027

Table 2: Test errors (absolute differences) for **synth** of different dropout methods and rates.



(a) Deep Ensembles (b) MC Dropout (c) SWAG

Figure 7: *Uncertainty tube* visualizations of one pathline from **synth** dataset that has a large variation at the end using three quantification methods. The seed location is at (-0.45,-0.1,-0.95).

We generated Fig. 6c to demonstrate the uncertainty estimated by the SWAG method. SWAG requires some additional training from a pre-trained model. Training 1000 steps using SGD takes 15 seconds on a NVIDIA 3090. Thus, we can quickly experiment with different hyperparameters in our applications.

Figure 6c shows that our SWAG model reports slightly higher uncertainty at the beginning (left) and lower uncertainty at the end compared to Deep Ensembles. However, the overall trend aligns with the result of Deep Ensembles. Especially, Fig. 7c shows that the uncertainty direction matches closer to the Deep Ensembles result, compared to the MC Dropout result.

In summary, SWAG gives a descent uncertainty quantification of the model, and it is easy to set up and use. Although it requires careful hyperparameter tuning, this process is not overly complicated. We recommend that users start with a rank of 100 and find the ideal swag_lr and n_swag_samples first, then adjust other parameters as needed.

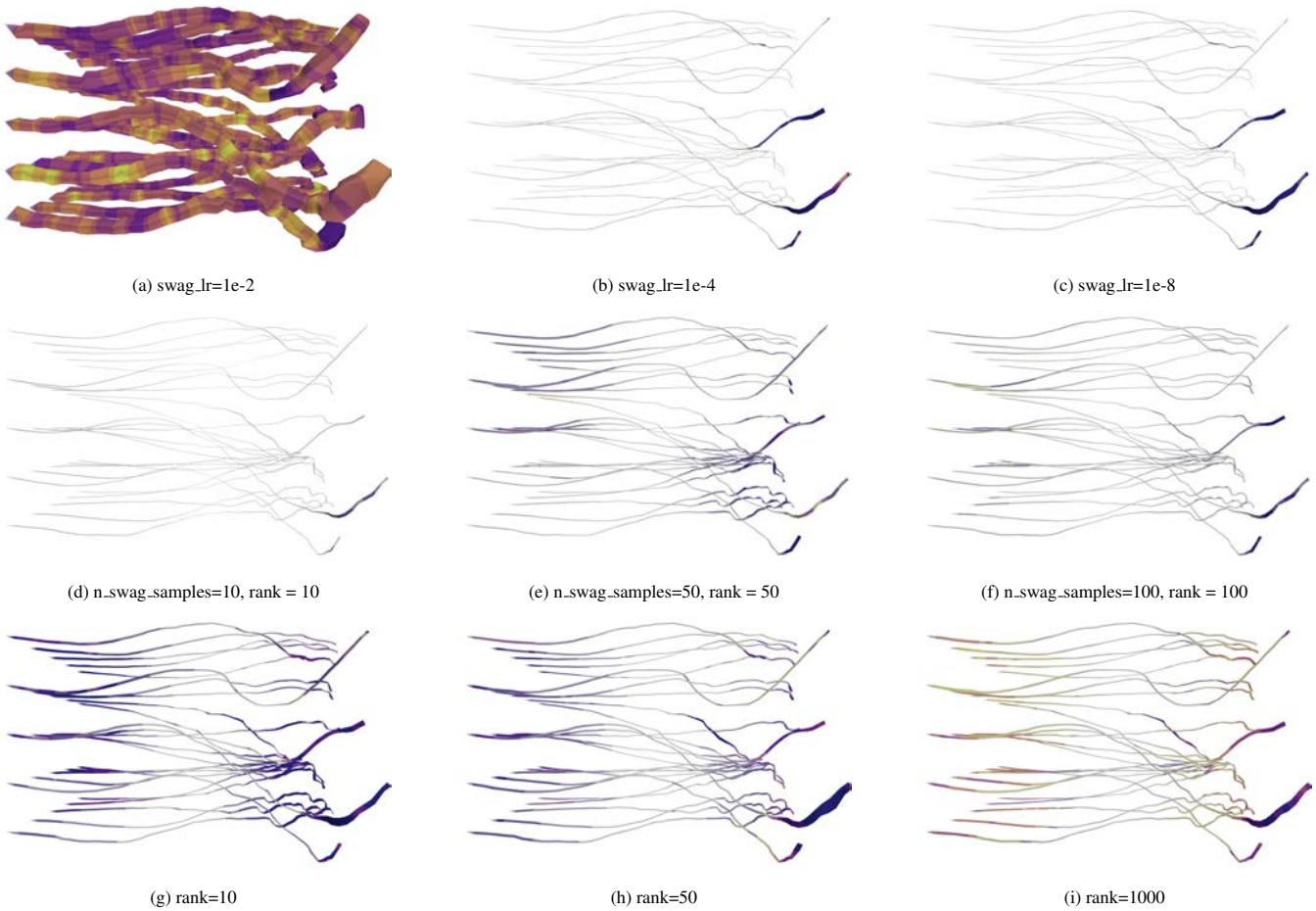
5.3.1 SWAG hyperparameter study

Here, we present a hyperparameter study for our **synth** model. In our implementation, we exposed five hyperparameters: swag_lr, n_swag_samples, rank, sgd_weight_decay, sgd_momemtum. The swag_lr controls the step size of the SGD in SWAG training. Maddox et al. [41] recommend a high constant learning rate so that SGD explores the model's weight space instead of simply converging to a local minimum. The number of steps SWAG training takes is the n_swag_samples. It needs to be set to a sufficiently large number to explore the weight space. And it heavily impacts the SWAG training time. Rank is the rank of the low-rank approximation of the covariance matrix of the multivariate Gaussian. The covariance matrix captures the correlation among the model parameters. Maddox et al. [41] stated that weight decay and momentum need to be explicitly specified. Then, SWAG can be viewed as a Bayesian inference approximation because weight decay with momentum corresponds to the prior distribution of the model weights.

We start from the base hyperparameter used in Fig. 6c: swag_lr=$5e-4$, n_swag_samples=1000, rank=100, sgd_weight_decay=$1e-8$, sgd_momemtum=0.9. We tested the $swag\_lr = [1e-2, 1e-4, 1e-8]$, $n\_swag\_samples = [10, 50, 100]$, $rank = [10, 50, 1000]$. For n_swag_samples, we also match the rank. The sgd_weight_decay and sgd_momemtum parameters are set to $1e-8$ and 0.9, respectively. We omit exploring these two parameters and recommend setting them according to the optimal training parameters.

The hyperparameter exploration results are presented in Fig. 8. For the swag_lr parameter, we recommend examining the global pattern of the network's predictions. That is, if uncertainty is globally high, we recommend reducing the swag_lr. In our test, we tested up to $1e-8$, and the SWAG samples still show reasonable uncertainty because we set n_swag_samples sufficiently large. n_swag_samples controls the number of steps SWAG takes in the model's weight space, and swag_lr controls the step size. The combination of those should be large enough to explore the weight space. Fig. 8d shows the classical underexplored weight space, showing no uncertainty in the model's prediction. The n_swag_samples parameter has a significant impact on the SWAG training time. In our experiments, the time it takes to train for 10, 50, 100, and 1000 steps is 1 second, 3 seconds, 6 seconds, and 15 seconds, respectively. We use a conservative setting of 1000 in this study.

As shown in the third row of Fig. 8, the effect of the rank parameter is not straightforward. The rank parameter sets the rank of the low-rank approximation of the covariance matrix of the model parameters. The consequence of the correlation between model parameters on the final prediction is not immediately clear to us. If the model is sufficiently small, we recommend examining the sin-



(a) swag_lr=1e-2 (b) swag_lr=1e-4 (c) swag_lr=1e-8

(d) n_swag_samples=10, rank = 10 (e) n_swag_samples=50, rank = 50 (f) n_swag_samples=100, rank = 100

(g) rank=10 (h) rank=50 (i) rank=1000

Figure 8: Hyperparameter test results of SWAG.

gular values obtained from the singular value decomposition of the full-rank covariance matrix and selecting the smallest reasonable rank for the low-rank approximation. In our experiment across different datasets and models, we found that 100 is a sufficiently large number. The rank also affects the SWAG training time and the time it takes to draw samples from the Gaussian. For ranks 10, 50, 500, 1000, the training takes 13, 13, 17, 29 seconds for 1000 n_swag_samples. Drawing 50 samples from the fitted Gaussian takes 32, 61, 320, and 700 milliseconds, respectively.

Across all hyperparameter tests, we compute the *uncertainty tube* using 25 seeds and 50 uncertainty samples, in addition to the original model's prediction. The model evaluation for the 25×51 pathlines takes 90 milliseconds, and the computation of the 25 uncertainty tubes takes approximately 350 milliseconds.

6 Results

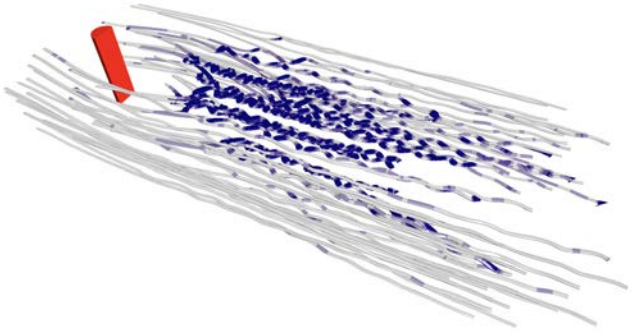
6.1 Tornado

This example utilizes flow maps generated from a synthetic **tornado** vector field dataset in Güther et al. [20]. We employ a flow map NN with four encoder layers, a 512 latent vector, and six decoder layers. The NN is trained on $131072 = 2^{17}$ trajectories integrated over 100 time steps with $\delta = 0.1$ and using Sobol seeds within subdomain $[-5,5] \times [-5,5] \times [-10,10]$. Our goal is to compare different visualization techniques for visualizing the model uncertainty calculated using the SWAG method.

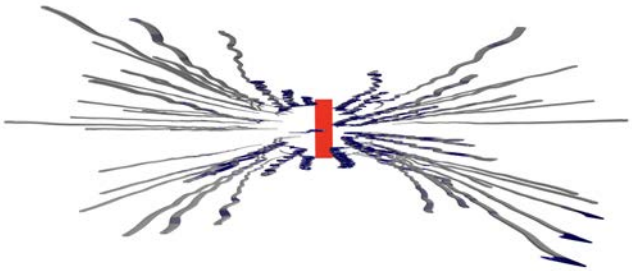
Figure 1 compares the spaghetti plot of ensemble members, the circular tube visualization, and the *uncertainty tube* for representing the model uncertainty. The ensemble visualization in Fig. 1.a leads to visual clutter that hinders the interpretation of the flow patterns. Additionally, the ensemble trajectories do not intuitively convey the model's uncertainty. The circular tube visualization in Fig. 1.b reduces the clutter and provides a better visual summary of the uncertainty compared to the spaghetti plot. However, it incorrectly assumes a symmetric distribution of trajectories around the mean. The *uncertainty tube* described in Sec. 4.2 encodes the variation direction using the major and minor axes of superellipses, as shown in Fig. 1.c. Our *uncertainty tube* enhances the distinction between the major and minor directions of variation and the visualization of how the direction changes along the trajectory integration, as depicted in boxes shown in Fig. 1. The *uncertainty tube* in the right box provides a more detailed representation, where changes in orientation and twisting along the trajectory are distinctly visible. In contrast, the circular tube on the left fails to capture or convey these variations. In cases of subtle changes along the trajectory, the color palette further enhances the distinction between low and high uncertainty, and between symmetric and nonsymmetric uncertainty. Overall, the *uncertainty tube* yields superior results in estimating and visualizing model uncertainty compared to the other approaches.

6.2 Half cylinder

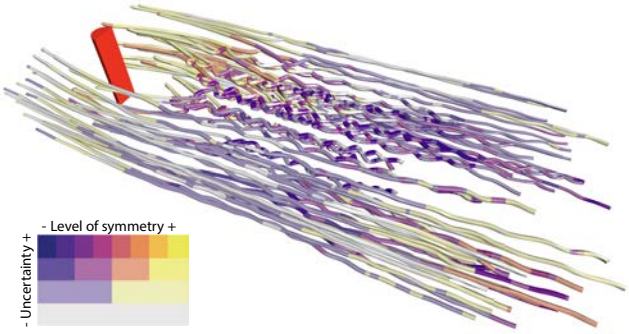
The **half cylinder** dataset is a time-varying flow field simulating flow over a half cylinder towards the positive x-direction. Visualizing asymmetric uncertainty enabled us to train a more accurate



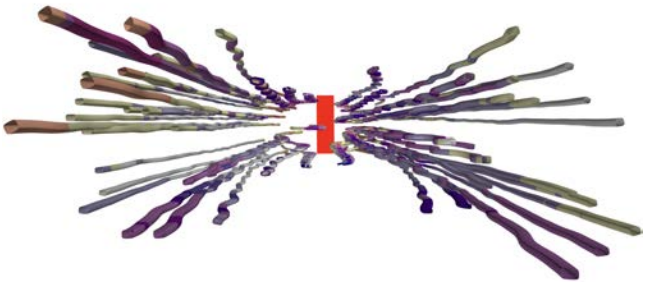
(a) Uncertainty from a model trained according to Han et al.[26].



(b) A view looking at yz-plane from +x.



(c) Uncertainty from a model trained with spatially uniform scaling.



(d) A view looking at yz-plane from +x.

Figure 9: Uncertainty Tube visualization of **half cylinder** dataset.

model.

In Han et al. [26], the data is mapped into a $[-1,1]^3$ space by rescaling the bounding box of the data. In Fig. 9a, we notice that all the *uncertainty tubes* appear flat in the y-direction, as shown in Fig. 9b. We suspect the flatness is caused by nonspatially uniform scaling of the data. We rescaled the data according to the actual ratio in the domain, so $[-1,1]^3$ represents a cube of size 8^3 in the domain. Although the training box $[-1,1]^3$ now contains empty spaces, spatially uniform scaling has reduced the evaluation error from 0.0058 to 0.0047. The evaluation error represents the absolute difference between the predicted location and the true location of the validation dataset in the original domain. We ran multiple random seeds to compare different training processes and consistently observed a reduction in the evaluation error. We also noticed a similar improvement by applying spatially uniform scaling of the **Hurricane** dataset used in Han et al. [26]. However, the generalizability of this finding to other MLP-based networks with sine activation is beyond the scope of this paper.

Another interesting effect is that, despite improved model quality, the amount of uncertainty (measured by the maximum eigenvalue) increased. This is illustrated by Fig. 9a showing more gray than Fig. 9c. Throughout our experiment, we consistently observed a mismatch between the amount of error and the model's uncertainty estimation. This mismatch means that extra caution is required when interpreting the results of uncertainty estimation, as they do not always reflect the actual model error but the model's confidence in its prediction.

7 Discussion

This paper introduced the *uncertainty tube*, a novel and computationally efficient visualization method for representing prediction uncertainty in neural network-derived particle trajectories. We design and implement a superelliptical tube that uniquely captures and intuitively conveys asymmetric uncertainty, thereby overcoming the limitations of conventional methods that typically assume symmetric uncertainty bounds. By integrating well-established uncertainty quantification techniques, including Deep Ensembles, MC Dropout, and SWAG, we demonstrated that the *uncertainty tube* significantly improves the representation of asymmetric uncertainty compared to the circular tube. Moreover, its rectangular design distinctly shows the uncertainty orientation and its evolution along the pathline more effectively than the circular and elliptical tubes. Our VSUP-inspired color map further helps distinguish different types of uncertainty when visualizing 3D geometries. In addition, we demonstrated one use case where we utilize asymmetric uncertainty to enhance training. We hope to explore additional ways to utilize uncertainty information to better understand the data, training, and models in the future.

The *uncertainty tube* visualization of the trajectory has some limitations. For example, when constructing the *uncertainty tube*, we projected the point at each step onto the plane orthogonal to the mean trajectory. This process eliminates the uncertainty along the mean trajectory. We could use color or texture to represent that type of uncertainty, or use local superquadric glyphs [63] to characterize the variation of all directions. Here, we focus on developing a visual representation that highlights the asymmetric nature of uncertainty in NN-based trajectories. A user-based study is necessary to evaluate the effectiveness and expressiveness of visual encoding for future research.

An important next step involves investigating more comprehensive uncertainty quantification methods, such as fully modeled Bayesian networks, especially to investigate how uncertainty is propagated through the data analysis pipeline. The way we utilized the three UQ methods in this paper assumes no uncertainty in the training data, which is rarely true in real-world applications.

Acknowledgments

This work was partially supported by the Intel OneAPI CoE, the Intel Graphics and Visualization Institutes of XeLLENCE, and the DOE Ab-initio Visualization for Innovative Science (AIVIS) grant 2428225.

References

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. doi: 10.1016/j.inffus.2021.05.008 2

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

[3] A. Agranovsky, H. Obermaier, C. Garth, and K. I. Joy. A Multi-Resolution Interpolation Scheme for Pathline Based Lagrangian Flow Representations. In *Visualization and Data Analysis 2015*, vol. 9397, p. 93970K, 2015. 2

[4] A. Arzani, J.-X. Wang, M. S. Sacks, and S. C. Shadden. Machine learning for cardiovascular biomechanics modeling: challenges and beyond. *Annals of Biomedical Engineering*, 50(6):615–627, 2022. 2

[5] K. Bao, X. Zhang, W. Peng, and W. Yao. Deep learning method for super-resolution reconstruction of the spatio-temporal flow field. *Advances in Aerodynamics*, 5(1):19, 2023. 2

[6] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1613–1622. JMLR.org, 2015. 3

[7] R. Bujack and K. I. Joy. Lagrangian Representations of Flow Fields with Parameter Curves. In *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 41–48. IEEE, 2015. 2

[8] G. Calzolari and W. Liu. Deep learning to replace, improve, or aid cfd analysis in built environment applications: A review. *Building and Environment*, 206:108315, 2021. doi: 10.1016/j.buildenv.2021.108315 2

[9] J. Chandler, H. Obermaier, and K. I. Joy. Interpolation-Based Pathline Tracing in Particle-Based Flow Visualization. *IEEE transactions on visualization and computer graphics*, 21(1):68–80, 2014. 2

[10] M. Correll, D. Moritz, and J. Heer. Value-suppressing uncertainty palettes. In *ACM Human Factors in Computing Systems (CHI)*, 2018. 5

[11] M. V. Da Costa and B. Blanke. Lagrangian methods for flow climatologies and trajectory error assessment. *Ocean Modelling*, 6(3-4):335–358, 2004. 2

[12] X. Dong and C. C. Hayes. Uncertainty visualizations: Helping decision makers become more aware of uncertainty and its implications. *Journal of Cognitive Engineering and Decision Making*, 6(1):30–56, March 2012. 3

[13] F. Ferstl, K. Bürger, and R. Westermann. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):767–776, 2016. doi: 10.1109/TVCG.2015.2467204 3

[14] L. L. Folgoc, V. Baltatzis, S. Desai, A. Devaraj, S. Ellis, O. E. M. Manzanera, A. Nair, H. Qiu, J. Schnabel, and B. Glocker. Is mc dropout bayesian? *arXiv preprint arXiv:2110.04286*, 2021. 3, 6

[15] G. Froyland and O. Junge. Robust FEM-Based Extraction of Finite-Time Coherent Sets Using Scattered, Sparse, and Incomplete Trajectories. *SIAM Journal on Applied Dynamical Systems*, 17(2):1891–1924, 2018. 2

[16] G. Froyland and K. Padberg-Gehle. 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*, 25(8):087406, 2015. 2

[17] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. F. Balcan and K. Q. Weinberger, eds., *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, New York, New York, USA, 20–22 Jun 2016. 2, 3, 4, 6

[18] M. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022. doi: 10.1016/j.engappai.2022.105151 2

[19] H. Gao, L. Sun, and J.-X. Wang. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, 33(7):073603, 2021. 2

[20] T. Günther, C. Rössl, and H. Theisel. Opacity optimization for 3d line fields. *ACM Trans. Graph.*, 32(4), July 2013. doi: 10.1145/2461912.2461930 8

[21] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J.-X. Wang, and C. Wang. SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 71–80. IEEE Computer Society, 2020. 2

[22] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller. A Critical Comparison of Lagrangian Methods for Coherent Structure Detection. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(5):053104, 2017. 2

[23] J. Han, J. Tao, and C. Wang. FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces. *IEEE transactions on visualization and computer graphics*, 26(4):1732–1744, 2018. 2

[24] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang. Flow Field Reduction Via Reconstructing Vector Data From 3-D Streamlines Using Deep Learning. *IEEE computer graphics and applications*, 39(4):54–67, 2019. 2

[25] J. Han and C. Wang. Tsr-vfd: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics*, 103:168–179, 2022. 2

[26] M. Han, J. Li, S. Sane, S. Gupta, B. Wang, S. Petruzza, and C. R. Johnson. Interactive Visualization of Time-Varying Flow Fields Using Particle Tracing Neural Networks . In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 52–61. IEEE Computer Society, Los Alamitos, CA, USA, Apr. 2024. doi: 10.1109/PacificVis60374.2024.00015 2, 3, 5, 9

[27] M. Han, S. Sane, and C. R. Johnson. Exploratory Lagrangian-Based Particle Tracing Using Deep Learning. *Journal of Flow Visualization and Image Processing*, 2022. doi: 10.1615/JFlowVisImageProc.2022041197 2, 3

[28] C. D. Hansen, M. Chen, C. R. Johnson, A. E. Kaufman, and H. Hagen. *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Springer Publishing Company, Incorporated, 2014. 3

[29] K. Höhlein, M. Kern, T. Hewson, and R. Westermann. A comparative study of convolutional neural network models for wind field downscaling. *Meteorological Applications*, 27(6):e1961, 2020. 2

[30] F. Hong, J. Zhang, and X. Yuan. Access Pattern Learning with Long Short-Term Memory for Parallel Particle Tracing. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 76–85. IEEE, 2018. 2

[31] J. Jakob, M. Gross, and T. Günther. A Fluid Flow Data Set for Machine Learning and its Application to Neural Flow Map Interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1279–1289, 2020. 2

[32] C. Johnson and A. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003. doi: 10.1109/MCG.2003.1231171 3

[33] A. Kamal, P. Dhakal, A. Y. Javaid, V. K. Devabhaktuni, D. Kaur, J. Zaientz, and R. Marinier. Recent advances and challenges in uncertainty visualization: a survey. *Journal of Visualization*, 24(5):861–890, Oct 2021. doi: 10.1007/s12650-021-00755-1 3

[34] A. Kumar, S. Garg, and S. Dutta. Uncertainty-aware deep neural representations for visual analysis of vector field data. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1343–1353, 2025. doi: 10.1109/TVCG.2024.3456360 2, 3, 4, 6

[35] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6405–6416. Curran Associates Inc.,

Red Hook, NY, USA, 2017. 2, 4

[36] J.-Y. Lee and J. Park. Deep Regression Network-Assisted Efficient Streamline Generation Method. *IEEE Access*, 9:111704–111717, 2021. 2

[37] Y. Li, C. Wang, and C.-K. Shene. Extracting Flow Features via Supervised Streamline Segmentation. *Computers & Graphics*, 52:79–92, 2015. 2

[38] M. Lino, S. Fotiadis, A. A. Bharath, and C. D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2275):20230058, 2023. doi: 10.1098/rspa.2023.0058 2

[39] C. Liu, R. Jiang, D. Wei, C. Yang, Y. Li, F. Wang, and X. Yuan. Deep Learning Approaches in Flow Visualization. *Advances in Aerodynamics*, 4(1):1–14, 2022. 2

[40] D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992. doi: 10.1162/neco.1992.4.3.448 3

[41] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pp. 13153–13164, 2019. 2, 3, 4, 6, 7

[42] M. Mirzargar, R. T. Whitaker, and R. M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2654–2663, 2014. doi: 10.1109/TVCG.2014.2346455 3

[43] M. Otto, T. Germer, H.-C. Hege, and H. Theisel. Uncertain 2D Vector Field Topology. *Computer Graphics Forum*, 2010. doi: 10.1111/j.1467-8659.2009.01604.x 3

[44] M. Otto, T. Germer, and H. Theisel. Uncertain topology of 3d vector fields. In *2011 IEEE Pacific Visualization Symposium*, pp. 67–74, 2011. doi: 10.1109/PACIFICVIS.2011.5742374 3

[45] T. A. J. Ouermi, J. Li, Z. Morrow, B. Van Bloemen Waanders, and C. R. Johnson. Glyph-Based Uncertainty Visualization and Analysis of Time-Varying Vector Fields . In *2024 IEEE Workshop on Uncertainty Visualization: Applications, Techniques, Software, and Decision Frameworks*, pp. 73–77. IEEE Computer Society, Los Alamitos, CA, USA, Oct. 2024. doi: 10.1109/UncertaintyVisualization63963.2024.00014 3

[46] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, Nov 1997. doi: 10.1007/s003710050111 3

[47] P. Pant, R. Doshi, P. Bahl, and A. Barati Farimani. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics of Fluids*, 33(10):107101, 10 2021. doi: 10.1063/5.0062546 2

[48] K. Potter, P. Rosen, and C. R. Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In A. M. Dienstfrey and R. F. Boisvert, eds., *Uncertainty Quantification in Scientific Computing*, pp. 226–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 3

[49] X. Qin, E. van Sebille, and A. S. Gupta. Quantification of errors induced by temporal resolution on lagrangian particles in an eddy-resolving model. *Ocean Modelling*, 76:20–30, 2014. 2

[50] R. Rahaman and a. thiery. Uncertainty quantification and deep ensembles. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., *Advances in Neural Information Processing Systems*, vol. 34, pp. 20063–20075. Curran Associates, Inc., 2021. 2

[51] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045 2

[52] T. Rapp, C. Peters, and C. Dachsbacher. Void-and-Cluster Sampling of Large Scattered Data and Trajectories. *IEEE transactions on visualization and computer graphics*, 26(1):780–789, 2019. 2

[53] J. Reyes, A. U. Batmaz, and M. Kersten-Oertel. Trusting ai: does uncertainty visualization affect decision-making? *Frontiers in Computer Science*, 7:1464348, 2025. 3

[54] M. P. Rockwood, T. Loiselle, and M. A. Green. Practical concerns of implementing a finite-time lyapunov exponent analysis with under-resolved data. *Experiments in Fluids*, 60(4):1–16, 2019. 2

[55] S. Sahoo and M. Berger. Integration-Aware Vector Field Super Resolution. 2021. 2

[56] S. Sahoo, Y. Lu, and M. Berger. Neural flow map reconstruction. *Computer Graphics Forum*, 41(3):391–402, 2022. doi: 10.1111/cgf.14549 2

[57] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In H. D. III and A. Singh, eds., *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 13–18 Jul 2020. 2

[58] S. Sane, R. Bujack, and H. Childs. Revisiting the Evaluation of In Situ Lagrangian Analysis. In *EGPGV@ EuroVis*, pp. 63–67, 2018. 2

[59] S. Sane, R. Bujack, C. Garth, and H. Childs. A Survey of Seed Placement and Streamline Selection Techniques. In *Computer Graphics Forum*, vol. 39, pp. 785–809. Wiley Online Library, 2020. 2

[60] S. Sane and H. Childs. Exploratory Time-Dependent Flow Visualization via In Situ Extracted Lagrangian Rßepresentations. In *In Situ Visualization for Computational Science*, pp. 91–109. Springer, 2022. 2

[61] S. Sane, C. R. Johnson, and H. Childs. Investigating In Situ Reduction via Lagrangian Representations for Cosmology and Seismology Applications. In *International Conference on Computational Science*, pp. 436–450. Springer, 2021. 2

[62] K. L. Schlueter-Kuck and J. O. Dabiri. Coherent structure colouring: identification of coherent structures from sparse data using graph theory. *Journal of Fluid Mechanics*, 811:468–486, 2017. 2

[63] T. Schultz and G. L. Kindlmann. Superquadric glyphs for symmetric second-order tensors. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1595–1604, 2010. doi: 10.1109/TVCG.2010.199 9

[64] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020. doi: 10.2514/1.J058291 2

[65] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, p. 681–688. Omnipress, Madison, WI, USA, 2011. 3

[66] C. Wittenbrink, A. Pang, and S. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):266–279, 1996. doi: 10.1109/2945.537309 3