An Interpolation Scheme for VDVP Lagrangian Basis Flows

Sudhanshu Sane[†1], Hank Childs[‡1] and Roxana Bujack[§2]

[1]Department of Computer and Information Science, University of Oregon, USA
[2]Los Alamos National Laboratory, USA

Abstract

Using the Eulerian paradigm, accurate flow visualization of 3D time-varying data requires a high temporal resolution resulting in large storage requirements. The Lagrangian paradigm has proven to be a viable in situ-based approach to tackle this large data visualization problem. However, previous methods constrained the generation of Lagrangian basis flows to the special case of fixed duration and fixed placement (FDFP), in part because reconstructing the flow field using these basis flows is trivial. Our research relaxes this constraint, by considering the general case of variable duration and variable placement (VDVP) with the goal of increasing the amount of information per byte stored. That said, reconstructing the flow field using VDVP basis flows is non-trivial; the primary contribution of our work is a method we call VDVP-Interpolation which solves this problem. VDVP-Interpolation reduces error propagation and limits interpolation error while using VDVP Lagrangian basis flows. As a secondary contribution of the work, we generate VDVP basis flows for multiple data sets and demonstrate improved accuracy-storage propositions compared to previous work. In some cases, we demonstrate up to 40-60% more accurate pathline calculation while using 50% less data storage.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

The analysis and visualization of time-varying flow phenomena introduces new challenges given the large amounts of data produced by CFD simulations on modern day supercomputers [ASM*11]. Given that improved computational capabilities have resulted in more data being produced faster than can be saved to disk, i.e., the I/O bottleneck, scientists have resorted to saving vector field time slices infrequently. However, accurate reconstruction of the flow field requires a high temporal resolution when using the Eulerian paradigm. This creates a tradeoff centered around large data and accuracy — saving frequently results in accurate interpolation, but creates a large data problem, while saving infrequently mitigates the large data problem, but is inaccurate. Agranovsky et al. [ACG*14] presented an alternative, Lagrangian-based method that improved the tccuracy-storage radeoffs. Their method has two phases, with the first phase extracting flow map samples or *basis flows* in situ, and the second phase involving interpolation using the extracted information to reconstruct the flow field. With our work, we improve on the Agranovsky method, thus further alleviating a large data visualization problem.

We contribute a new interpolation scheme to consume information extracted in situ which enables new techniques for the Lagrangian paradigm to maintain high accuracy while reducing data storage. Previous work considered using basis flows of fixed duration and fixed placement (FDFP). However, post hoc reconstruction of the flow field using short basis flows, while relatively straightforward, results in inaccuracy due to "stitching" a particle trajectory together [BJ15, HBJG16]. Each stitching event corresponds to a particle basis flow neighborhood update and propagates a local truncation error. Our work introduces the notion of variable duration and variable placement (VDVP) Lagrangian basis flows which enables the use of longer basis flow trajectories. We introduce VDVP-Interpolation, an interpolation scheme that can use longer Lagrangian basis flows to calculate new particle trajectories with reduced error propagation and accumulation. The VDVP-Interpolation scheme allows particles to maintain their basis flow neighborhoods for longer durations, i.e., fewer "stitching" events, and limits interpolation error by evaluating the particle neighborhood.

Our research furthers the usage of Lagrangian analysis as a reduction operator for time-varying flow data by increasing the information content per byte. The use of VDVP allows for much variation in the specific placement and durations of extracted basis flows, potentially allowing for saving more information per byte than FDFP. Further, it enables reduced error propagation from the use of longer trajectories. To realize the benefits of VDVP, an in-

† ssane@uoregon.edu
‡ hank@uoregon.edu
§ bujack@lanl.gov

terpolation scheme that makes optimal usage of such input is necessary. This paper contributes that component, i.e., an interpolation scheme for VDVP Lagrangian basis flows, enabling future in situ methods research. Our evaluation is aimed at demonstrating the value of the VDVP approach, and thus the value of our interpolator. We consider multiple data sets and demonstrate improved accuracy-storage propositions compared to previous methods. As a result of using both VDVP-Interpolation and VDVP Lagrangian basis flows, we calculate more accurate pathlines while using less data storage.

Our specific contributions with this work are:

- We contribute a configurable, neighborhood-aware interpolation scheme for Lagrangian basis flows that vary in seed position and duration.
- Building on previous theoretical work, we present the first implementation of generating and using basis flows of variable duration and variable placement (VDVP), forming a foundation for future research.
- We demonstrate better accuracy-storage propositions compared to previous work.

2. Background and Related Work

2.1. The Lagrangian Frame of Reference

The Lagrangian frame of reference describes a flow parcel as it moves through space and time in the flow field. Information is stored in the form of a flow map when using the Lagrangian frame of reference. 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 particle starting at position $x_0 \in \mathbb{R}^d$ and time $t_0 \in \mathbb{R}$ moves to in the time interval $[t_0, t] \subset \mathbb{R}$. In contrast to the Eulerian approach which stores a time slice, the stored information represents a time interval and new particle trajectories can be computed through interpolation using the basis of known trajectories.

2.2. Flow Analysis using Lagrangian Techniques

Particle trajectories are one of the fundamental elements of flow visualization [LHD*04, MLP*09, PPF*11]. Various works have considered Lagrangian-based flow information storage and pathline construction techniques. Hlawatsch et al. [HSW11] focused on precomputing Lagrangian-based trajectories and optimally selecting which to use when calculating pathlines via a hierarchical scheme. Agranovsky et al. [AGJ11] studied the use of Moving Least Squares and Barycentric coordinate interpolation, to optimize pathline interpolation using scattered particles. Sauer et al. [SXM16] presented a new joint data representation which combines the Eulerian and Lagrangian reference frames. Chandler et al. [COJ15] proposed a modified k-d tree to store particle locations at a given time and an associated interpolation scheme for SPH [GM77] simulations. In the context of reducing storage and error for in situ supercomputing environments, Agranovsky et al. [ACG*14] used a flow map over a vector field to represent a flow and performed barycentric coordindate interpolation post hoc (details in Section 2.5). This study was extended to provide a better understanding of spatiotemporal trade-offs [SBC18]. In this work, we use the method proposed by Agranovsky et al. [ACG*14] as a baseline for comparison.

With a focus on identifying sources of error in Lagrangian-based advection methods Chandler et al. [CBJ16] showed the correlation of error with divergence for their interpolation-based pathline tracing system. Bujack et al. [BJ15] identified neighborhood updates as the source of error propagation. Hummel et al. [HBJG16] theoretically extended this work by using upper error bounds to visualize the uncertainty of the pathlines.

2.3. Seed Placement Techniques for Flow Analysis

Several works have presented seed point placement and streamline selection algorithms to explore flow fields [ZSH96, JL97, VKP00, MTHG03, LMG06, RPP*09, XLS10, WLZMI10]. However, the majority of these works deal with steady state flow. Seed point placement strategies to extract information and maintain coverage of a 3D time-varying flow is limited [MLP*10]. Specifically related to strategies for the extraction of basis flows, Agranovsky et al. [ACG*14] placed seeds along a uniform grid periodically. With our work, in addition to primarily allowing particles to follow the flow, we strategically introduce basis flow seeds to limit the error during post hoc flow field reconstruction and maintain an approximately uniform particle distribution over time (details in Section 5). The system we adopt is most similar to Mebarki et al. [MAD05] who used Delaunay triangulation to identify cavities in the field and then placed seeds at the centroid of the triangle.

2.4. In Situ Processing

An emerging paradigm to counteract temporal sparsity is the use of in situ processing [BAA*16]. In situ processing operates as the simulation produces data, giving it the significant advantage of access to all of the simulation data, i.e., the complete spatial data at full temporal resolution. The Lagrangian paradigm is well suited for in situ processing because the basis of known trajectories, representing an interval of time, can be calculated accurately in situ, where all the simulation data is available. In contrast, storing a temporally sparse subset of the information in its Eulerian specification and then integrating post hoc results in significant approximation errors due to error propagation in the numerical integration. Thus, the Lagrangian representation has the potential to represent more information per byte stored.

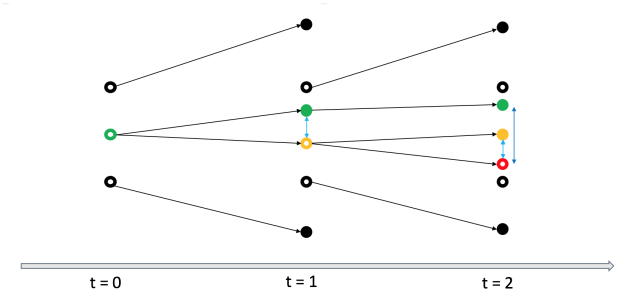
2.5. State of In Situ Lagrangian Techniques

Agranovsky et al. [ACG*14] presented an approach that is useful for exploratory flow analysis, i.e., analysis when the user does not know which particle trajectories are desired before the simulation is run. In the first phase, basis flows are calculated in batches in situ. Particles are seeded along a uniform grid to begin a batch. These particles advect for a fixed number of cycles (e.g., 200 cycles), to form basis flows. The particles are then terminated and the end points of the basis flows are stored to disk. The cycle when data is stored to disk is referred to as a "write cycle." The process then repeats until the simulation completes.

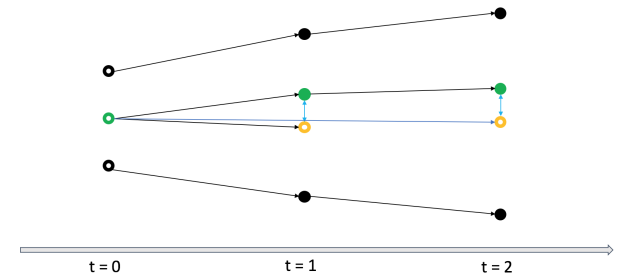
In the second phase, the basis flows from the first phase are used to approximate the behavior of the flow field. To begin, for a given particle, the algorithm identifies a neighborhood of surrounding basis flows to follow. Specifically, the neighborhood is the set of basis flows that form a minimum convex hull around the particle in space

and time. The particle's next position is determined by interpolating the basis flows via barycentric coordinate interpolation. This process advances the particle to the same time as when the current batch of basis flows ends. To advance the particle further, the process is repeated with the following batches of basis flows, until the particle reaches its desired termination time. Agranovsky et al.'s study showed that using the Lagrangian approach is significantly superior to the Eulerian approach under sparse temporal settings. Agranovsky's seminal work falls in the FDFP (fixed duration, fixed placement) category of basis flows. We refer to the associated interpolation scheme using FDFP basis flows as **FDFP-Interpolation**.

3. Motivation



(a) *Basis flows are plotted in black, with the basis flow seed being a hollow black circle and the basis flow end point being a solid black circle. The desired trajectory to interpolate starts at the hollow green circle. The hollow yellow and hollow red circles are the interpolated positions from using short basis flows. In this case, the slightly incorrect position from the interpolation error at $t = 1$ (hollow yellow) leads to an even more incorrect position at $t = 2$ (hollow red), i.e., error propagation. The solid green and solid yellow are the correct particle end locations for each respective interpolation. The relatively small local error (distance between solid green and hollow yellow, or solid yellow and hollow red) is $(\frac{1}{2}h_x^2\|f''\|)$ [BJ15]. The local error propagates with each interpolation. The global error is enhanced by the Lipschitz constant $h_t L$ of f. Thus, at $t = 2$, the global error is already $\frac{1}{2}h_x^2\|f''\|(1+h_t L)$ [HBJG16].*



(b) *Interpolation error when using longer basis flows. The local interpolation error for each step is inevitable, but using the original neighborhood prevents the incorrect intermediate results from influencing the future path of the particle. The overall global error is then limited to the local interpolation error $\frac{1}{2}h_x^2\|f''\|$.*

Figure 1: *A notional example to provide intuition of how longer basis flows can reduce error propagation.*

Problem: The FDFP-Interpolation approach can suffer from local truncation error propagation. A particle is advanced in time

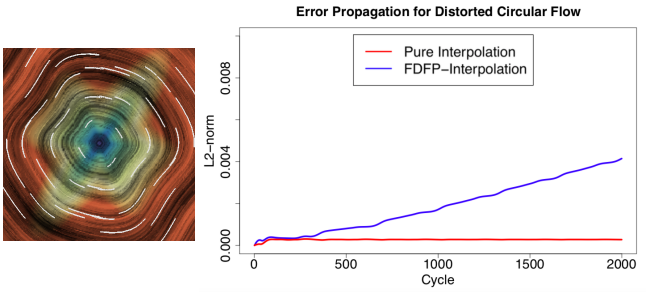


Figure 2: *Motivating result comparing FDFP-Interpolation to pure interpolation on an analytic data set of distorted circular flow data. The image on the left is the LIC of the flow field (color encodes the velocity magnitude). The white lines are the FDFP basis trajectories. The image on the right is a plot of error propagation over 2000 cycles. In contrast to FDFP-Interpolation, the pure interpolation only shows local interpolation error and has no error propagation.*

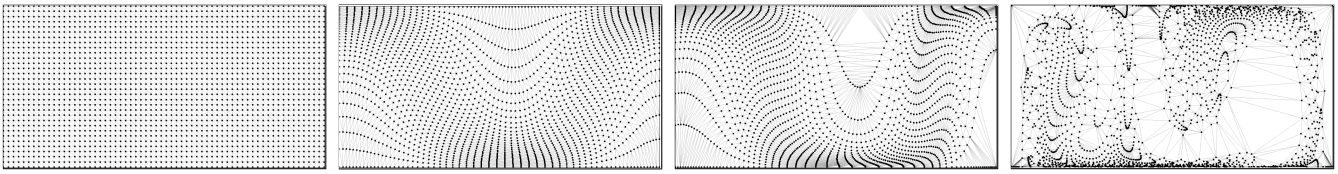
by following a neighborhood of basis flows. However, given that the basis flows are calculated in batches for the FDFP approach, the process requires identification of a new neighborhood, i.e., a neighborhood update, for each step (advancement in time). To produce the final particle trajectory, interpolation steps are stitched together as the particle is advanced forward in time. Figure 1a illustrates how a small local truncation error occurs with each interpolation step. Further, this local truncation error propagates with each interpolation step resulting in an increase of the global truncation error. The details of the error propagation and accumulation have been shown by Bujack et al. [BJ15]. The final accuracy is then dependent on the number of interpolation steps stitched together, i.e., the number of neighborhood updates. When the number of interpolation steps being stitched together is high, as in the case for long simulation runs, the error propagation and accumulation can grow exponentially and lead to poor accuracy [SBC18].

Our Solution: Extend the duration of basis flows for as long as possible. The error propagation and accumulation occurs for every instance of a stitching event (neighborhood update). We can mitigate this issue if:

1. Basis flows live for the duration of the simulation.
2. The interpolation is done based on the initial neighborhood information. Having basis flows live for the duration of the simulation means a particle can have the same neighborhood for each interpolation step.

Calculating a particle trajectory would then require only interpolation (i.e., from start time to current time using the same basis flows) and there would be no error propagation events since there is no need for a neighborhood update. Then, the error of this pure interpolation approach is $O(h_x^2)$, where h_x is the resolution in space [BJ15]. Figure 1b illustrates particle interpolation by using the same neighborhood.

The FDFP-Interpolation scheme suffers from local truncation error propagation, while our approach uses pure interpolation. To highlight the difference in error propagation and accumulation between the two methods, we consider an analytic field — a distorted circular flow. Figure 2 shows that the pure interpolation approach



(a) *Cycle - 0* (b) *Cycle - 100* (c) *Cycle - 200* (d) *Cycle - 1000*

Figure 3: *Particle distribution for the Double Gyre, with period set to 1000 cycles. Figure d shows significantly under and over represented regions of the flow.*

has absolutely no error propagation, while the stitching together of trajectories shows a growth in error for every advancement in time (cycle).

Problem: The interpolation error can become unbounded in divergent areas. While using longer basis flows for interpolation reduces error propagation, generating longer basis flows may result in certain regions having poor basis coverage, depending on the nature of the flow field. Figure 3 shows the distribution of particles at various stages when considering the Double Gyre [SLM05]. Figure 3a shows the initial distribution of particles along a uniform grid. We can observe the divergence of the particles in Figures 3b and 3c. There are observable regions in the field that are under and over represented in Figure 3d. If the basis flows of a neighborhood diverge, i.e., the neighborhood is stretched or basis flow particles separate, then the neighborhood size $h_x \in \mathbb{R}$ can become unbounded. Using the pure interpolation approach with divergent basis flows will result in a high linear interpolation error (with overall performance then being worse than FDFP-Interpolation). This is in accordance with Chandler et al. [CBJ16], who show the correlation between using diverging basis flows and post hoc interpolation error. If new particles are not frequently introduced, then the post hoc analysis of the underrepresented regions could be poor.

Our Solution: Extend the duration of basis flows for as long as possible, but update the particle neighborhood if it diverges beyond a limit. In this paper, we propose a hybrid approach between the uniform case and the pure interpolation approach. As input, we generate basis flows of variable duration and variable placement (VDVP) during the first phase. When performing interpolation using the VDVP Lagrangian basis flows, as long as a particle lives in a non-divergent neighborhood, it uses the pure interpolation approach. As soon as particle neighborhood divergence is detected, the particle neighborhood is updated. In order to guarantee that a small neighborhood can always be found, an approximately uniform distribution of particles is required in the domain. There are several ways in which this can be achieved. Our VDVP approach introduces new particles with the objective of limiting post hoc interpolation error.

The following sections provide details regarding the implementation of our solution. **VDVP-Interpolation** is our neighborhood-aware interpolation scheme for VDVP Lagrangian basis flows. It enables interpolation with reduced error propagation when using longer basis trajectories. Further, it limits interpolation error by evaluating particle neighborhoods for divergence and only updates if it exceeds a limit for h_x.

4. VDVP-Interpolation Method

Our Lagrangian-based technique is implemented by following the same high level approach, in that it is a two stage process, as described in Section 2.5. However, to effectively use longer duration basis flows we designed a simple interpolation scheme that evaluates the quality of the particle neighborhood formed by in situ extracted basis flows at each step.

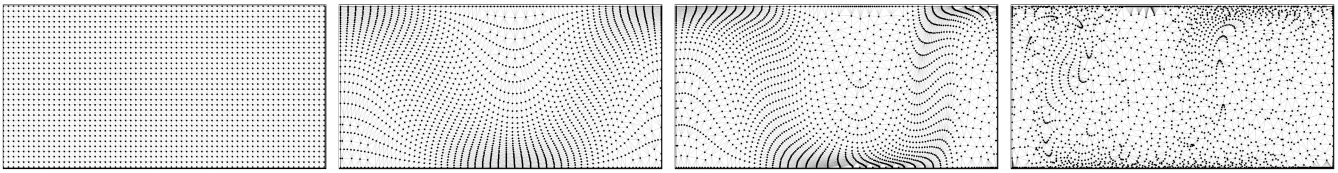
Given a set of VDVP input basis flows, such that each individual basis flow is represented as a starting location at time T_i, zero or more intermediate locations, and an end location at time T_{i+j}, where $j \geq 1$. A basis flow can exist for as short as a single step, or for as long as the length of the simulation. Additionally, there are no contraints on the spatial location of the basis flows. However, from a temporal perspective, locations of the basis flow are only stored at write cycles, i.e., T_i, T_{i+1}, ... , T_{i+j} correspond to times at write cycles. For a given particle location P_0 at time T_0, our interpolation scheme starts by identifying a neighborhood of basis flows B_1, B_2, ... , B_n (where $n = 3$ for 2D and $n = 4$ for 3D) surrounding P_0. Given a neighborhood of basis flows to follow, we interpolate each particle trajectory location using barycentric coordinates interpolation. In an ideal case, we can follow the same neighborhood of basis flows, performing each interpolation from the starting location, to calculate an entire particle trajectory with no error propagation.

To begin, an interpolation step is performed using the neighborhood of basis flows of P_0 at time T_0, to calculate the next location P_1 at time T_1. After the interpolation step, we evaluate the neighborhood of basis flows at T_1. We perform a neighborhood update if:

- **A basis flow B_i of the particle neighborhood terminates.** In this case we need to identify a new neighborhood of basis flows to continue particle trajectory interpolation.
- **Basis flows of particle neighborhood diverge.** We evaluate the neighborhood of basis flows to keep the interpolation error bounded. If the basis flows are deemed to have diverged, we perform a neighborhood update.

If a neighborhood update is not required, then we use the same neighborhood of basis flows of P_0 at time T_0, to calculate the next location P_2 at time T_2, i.e., a longer interpolation step is performed by following the same set of basis flows. The process is then repeated by evaluating the neighborhood of basis flows at time T_2 and so on.

If a neighborhood update is performed, then we use the new neighborhood of basis flows of P_1 at time T_1, to calculate the next location P_2 at time T_2. The process is then repeated by evaluating the neighborhood of basis flows at time T_2 and so on.



(a) *Cycle - 0* (b) *Cycle - 100* (c) *Cycle - 200* (d) *Cycle - 1000*

Figure 4: *Particle distribution after addressing under and over represented regions for the Double Gyre.*

To identify particle neighborhoods at time T_i, we first perform a single Delaunay triangulation over all basis flow particle locations at time T_i. If required, each particle neighborhood can then be identified as the cell containing the particle location P_i at time T_i.

A particle neighborhood is deemed to have diverged if the circumradius of the cell, representing the particle neighborhood, is greater than a user-defined parameter, $UpperThreshold$. Barycentric coordinates interpolation error is bounded from above through the circumradius $R \in \mathbb{R}$ of the corresponding cell. The interpolation error is given by the equation:

$$\|f(x) - Lf(x)\| \leq \frac{1}{2}R^2\|f''\|_\infty \tag{1}$$

where $f(x)$ is the ground truth location, $Lf(x)$ is the barycentric coordinates interpolated location, and $\|f''\|_\infty$ is the maximum function space norm of the second derivative of f [Wal98].

Our technique is configured to limit interpolation error by only using particle neighborhoods that have a circumradius less than $UpperThreshold$. In the following subsection we describe measures taken to generate VDVP basis flows that guarantee a neighborhood with circumradius less than $UpperThreshold$ can be found for each time step.

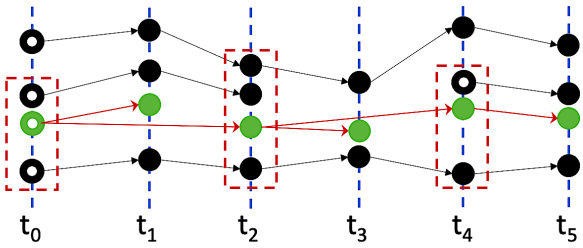


Figure 5: *A notional example of VDVP-Interpolation. Basis flows are plotted in black and a sample particle trajectory being interpolated is shown in green. Hollow circles are initial positions. Red arrows show interpolation. Red dashed boxes denote neighborhood update events. t_0 - particle identifies an initial neighborhood. t_1 - particle maintains neighborhood. t_2 - particle neighborhood update (neighborhood basis flow terminates). t_3 - particle maintains neighborhood. t_4 - particle neighborhood update (basis flows diverge).*

Data: ParticleSet P, BasisFlowSet B,
float $UpperThreshold$, int T_{start}, int T_{end},
int $WriteInterval$
Function VDVP-Interpolation()
$T_{current} = T_{start}$;
while $T_{current} < T_{end}$ **do**
  DT = Delaunay($B, T_{current}$);
  if $T_{current} = T_{start}$ **then**
    foreach *Particle* $p \in P$ **do**
      $p.NB$ = UpdateNBInfo(p,DT);
    end
  else
    foreach *Particle* $p \in P$ **do**
      if *EvaluateNB*($p.NB, UpperThreshold$) **then**
        $p.NB$ = UpdateNBInfo(p,DT);
      end
    end
  end
  foreach *Particle* $p \in P$ **do**
    p = Interpolate($p, p.NB$);
  end
  $T_{current} = T_{current} + WriteInterval$;
end

Algorithm 1: *VDVP-Interpolation Algorithm*

5. Generation of VDVP Basis Flows

In order to evaluate VDVP-Interpolation we need to generate VDVP basis flows. Our objectives are to generate long duration basis flows and simultaneously provide sufficient coverage to limit interpolation error. VDVP basis flow generation and distribution management can be guided by distance fields, spatial binning, neighborhood entropy, vector field divergence, and so on. Determining an optimal and efficient basis flow particle distribution approach in situ is a large topic beyond the scope of this paper and will be considered as future work. VDVP-Interpolation can be configured to interpolate VDVP basis flows, with any spatial distribution, if represented as defined in the previous section. For this study, we address the problem of underrepresented regions or particle clustering that come from allowing particles to follow longer trajectories, by employing a method to limit interpolation error.

With our VDVP approach, we begin by placing particles along a uniform grid in the volume. These particles are advected through the time steps until a write cycle completes. At the end of a write cycle, the particle positions are saved to disk. A particle is terminated if it exits the volume. Advection continues for the remaining particles from their last position. Thus, at write cycles (i.e., simulation cycles where data is saved to disk), intermediate locations along a particle trajectory are saved to disk. This results in longer basis flow trajectories with the distribution of seeds determined by

the flow itself. Figure 3 shows the distribution of particles achieved by a flow-guided VDVP approach for the Double Gyre data set.

To limit interpolation error, our goal is to guarantee a particle neighborhood update during interpolation can find a valid sized neighborhood. In addition to placing new seed particles to address the problem of underrepresented regions, we selectively terminate basis flows to mitigate particle clustering. To identify candidate regions for particle addition and removal, we perform Delaunay triangulation on the existing particles in the volume at the end of a write cycle. The circumradius of the largest Delaunay cell has a direct relationship to how sparse the particle sampling is in that region. The circumcenter is farthest away from any other current particle, and therefore a natural candidate to insert a seed to improve the overall coverage. If the circumradius of a cell is larger than $UpperThreshold$, we place a seed point at the circumcenter if it lies inside the cell, or at the location on the boundary of the cell that is closest to the circumcenter if it is outside. For particle removal, for every vertex in the triangulation we calculate the average circumradius of cells that the vertex is a member. If the average calculated circumradius is below a user-defined threshold $LowerThreshold$, the associated basis flow particle is removed.

Figure 4 shows a more uniform distribution of particles achieved by a flow-guided VDVP approach using particle distribution management for the Double Gyre data set.

6. Study Overview

We evaluate the VDVP-Interpolation method using results of the FDFP-Interpolation approach as a baseline for comparison. For our study, we generate our input basis flows by evaluating analytic data sets on the fly or loading simulation results that were precalculated for each cycle from disk. The study consists of configurations which vary over five parameters:

1. Lagrangian-based techniques
2. Data sets
3. Total data storage
4. Number of cycles saved (write cycles)
5. Number of basis flows saved per write cycle

We test our implementation on a single node. We ran a total of 144 test configurations on a Xeon E5-2667v3 CPU. We used 12 cores at 3.2GHz and 256 GB DDR4 memory. Post hoc particle interpolation and basis flows particle advection was performed in parallel using OpenMP. We used the CGAL library to calculate the Delaunay triangulation, and to perform vertex insertion and deletion.

6.1. Configuration Parameters

6.1.1. Lagrangian-based techniques

We compare the FDFP-Interpolation using FDFP input basis flows to VDVP-Interpolation using VDVP input basis flows. We generate multiple sets of each type of input basis flows by varying configurations parameters.

6.1.2. Data Sets

We considered three data sets to evaluate our method:

Double Gyre — This data set is an analytic two-dimensional flow field that is commonly used to study flow visualization techniques [SLM05]. It consists of two counter-rotating gyres with a time dependent perturbation. The data set is simulated for 2048 cycles at a base resolution of 512×256 ($\approx$ 6.4GB). We set the period of the Double Gyre flow to 1000 cycles (each cycle is 0.01 seconds).

Arnold-Beltrami-Childress (ABC) — This data set is a time-dependent variant of the three-dimensional ABC analytic vector field [BCT01]. For this variant of the ABC analytic vector field we used $A = B = C = 1$ and selected values of $\varepsilon = 1$ and $\Omega = 1$. The data set is simulated for 2048 cycles at a base resolution of $128 \times 128 \times 128$ ($\approx$ 103GB). We set the period of the ABC flow to 1000 cycles (each cycle is 0.001 seconds).

Tornado — This data set is a real-world simulation of the dynamics of an F5 tornado [OWW15]. The base resolution is $490 \times 490 \times 280$. A mature tornado vortex (depicted in Figure 6) exists in the domain during the 512 simulation seconds we considered for our experiments. Our collaborating scientist normally uses a frequency of “every two simulation seconds" to study this turbulent data set. Thus, we considered 257 time slices ($\approx$ 415GB), with the time-steps evenly distributed from $t_0 = 8502$s to $t_{256} = 9014$s.

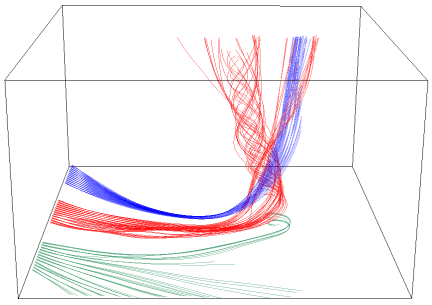


Figure 6: *Pathlines traced depict a mature tornado vortex.*

6.1.3. Total Data Storage, Number of Cycles Saved, and Number of Basis Flows Saved per Cycle

The total number of basis flows saved, i.e., the total data storage, is the summation of the number of basis flows saved over every write cycle. For FDFP input, the number of basis flows saved every write cycle can be fixed. Let P denote the number of basis flows saved at a write cycle. Let N_C denote the number of cycles saved, i.e., the number of write cycles. Then, the total data storage required can be calculated as the product of N_C and P. If X denotes the total data storage, then $X = N_C \times P$. We select multiple combinations of P and N_C for a given X. The selected combinations of P and N_C are together a set of configurations to generate FDFP basis flows. For the Double Gyre and ABC data set, $N_C = \{8,\ 16,\ 32,\ 64,\ 128,\ 256,\ 512,\ 1024,\ 2048\}$. For the Tornado data set, $N_C = \{8,\ 16,\ 32,\ 64,\ 128,\ 256\}$.

For our study, we have three sets of selected combinations of N_C and P, for the evaluation of the FDFP-Interpolation approach. The set of options for N_C remains the same across all three sets of selected combinations. The values of P in the second and third set are two times and four times the corresponding values of P in the

first set. Thus, the total data storage of the second and third set is two times and four times respectively. We denote these sets of test configurations as FDFP-1X, FDFP-2X, and FDFP-4X.

We selected the smallest value of total data storage, $1X$, used to calculate multiple combinations of P and N_C, to be equal to two times the total number of grid points in the base resolution of the specific data set. For the Double Gyre data set, $X = (512 \times 256) \times 2 = 262,144$ points ($\approx$ 6MB). For the ABC data set, $X = (128 \times 128 \times 128) \times 2 = 4.2$M points ($\approx$ 100MB). For the Tornado data set, $X = (490 \times 490 \times 280) \times 2 = 134.4$M points ($\approx$ 3.2GB).

When generating VDVP input, the number of basis flows fluctuates over time, i.e., the number of basis flows saved every write cycle is not fixed. Thus, in the case of our VDVP input, the total data storage is observed. To compare VDVP-Interpolation with FDFP-Interpolation, we have a corresponding set of test configurations, with the same three sets of selected combinations of P and N_C to generate VDVP basis flows. However, the value of P is only the initial number of basis flows placed, i.e., it is not fixed. We denote these corresponding sets of test configurations as VDVP-1X, VDVP-2X, and VDVP-4X.

In addition to particles exiting the domain, the total data storage costs of VDVP is influenced by particle addition and removal. Let R denote the circumradius of a cell after the initial placement of particles along a uniform grid. Then, we define $UpperThreshold$ and $LowerThreshold$ as follows:

$$UpperThreshold = CR \tag{2}$$

$$LowerThreshold = \frac{R}{C} \tag{3}$$

where C is a user-defined value to control particle addition and removal. For our study, we empirically selected $C = 2$ for the two-dimensional Double Gyre data set, and $C = 8$ for the three-dimensional ABC and Tornado data sets. We found these values allowed us to keep particle addition and removal relatively balanced.

6.2. Error Evaluation

We calculate particle trajectories using three methods.

- **Ground Truth** — The particle trajectory is calculated with a fourth-order Runge Kutta scheme [CK90] using the full spatial and temporal resolution available. The ground truth is considered to be perfectly accurate, i.e., it has 0% error.
- **FDFP-Interpolation** — Lagrangian particle trajectories are calculated by FDFP-Interpolation using FDFP input basis flows for every configuration of N_C and P.
- **VDVP-Interpolation** — Lagrangian particle trajectories are calculated by VDVP-Interpolation using VDVP input basis flows for every configuration of N_C and P. In this case, P is only the initial number of basis flow particles seeded in the volume.

We evaluate the accuracy of the Lagrangian particle trajectories calculated from a test configuration by comparing it to the calculated ground truth. For both the Double Gyre and ABC data set we randomly seed 1000 particles in the volume. For the Tornado data set, we place 144 particles along rakes at locations used by our collaborating scientist to study the phenomena (Figure 6). We then calculate the set of trajectories for each test configuration.

To compare two trajectories we measure the L2-norm. N_C is the number of cycles saved and consequently the number of known particle positions along a Lagrangian trajectory.

The average L2-norm is calculated as follows —

$$\frac{1}{p}\sum_{i=0}^{p}\frac{1}{N_C}\sum_{t=0}^{N_C}||x_{i,t} - g_{i,t}|| \tag{4}$$

where p is the total number of particles, $x_{i,t}$ is the location of a Lagrangian interpolated particle i at time t and $g_{i,t}$ is the location of the ground truth particle i at time t.

Thus, we evaluate the distance between the ground truth and a Lagrangian trajectory at every known point of the Lagrangian trajectory. The points that are known of the Lagrangian trajectory can be connected using linear interpolation or curve fitting. Representation of a Lagrangian trajectory using curve-fitting has been studied by Bujack et al. [BJ15]. For our study, we focus on the accuracy of the interpolated locations of a Lagrangian particle trajectory.

7. Results

The accuracy of interpolated pathlines is dependent on both the input basis flows and the interpolation scheme used. VDVP-Interpolation can utilize the FDFP input and produce pathlines of the same accuracy as FDFP-Interpolation. Given both approaches require varying data storage, we take the number of basis flows used for the pathline interpolation into account.

7.1. Accuracy and Data Storage Comparison

We analyze the results of accuracy achieved and the corresponding number of basis flows used by each approach, shown in Figure 7. The x-axis represents the average number of points stored per write cycle, denoted by P_{avg}, and uses a logarithmic scale. The y-axis represents the average L2-norm and uses a linear scale. Each curve represents one of the sets of configurations. Thus, for each curve, as the number of particle locations saved increases (P_{avg}), the number of cycles (N_C) saved decreases. Further, since number of basis flows used does not match exactly for corresponding configurations of each approach, we highlight example configurations which used approximately similar amounts of data storage or achieved similar accuracy for different amounts of data storage.

For the Double Gyre plot in Figure 7a, considering all configurations, VDVP-Interpolation calculated particle trajectories that are 48% more accurate on average than the corresponding FDFP-Interpolation approach, however, it used 16% more data storage on average. Our VDVP generation approach produced more basis flows given the divergent nature of the flow and the contained nature of the data set, i.e., no particles exit. We observed the number of basis flows generated was proportional to the value of N_C for this data set. As the interval increases, i.e., as N_C gets smaller, and the opportunities to add particles coincides with clustering of particles in the domain, we observe a more balanced particle addition and removal. For configurations between $N_C = 64$ and $N_C = 8$, we observe VDVP-Interpolation is on average 50% more accurate while using 2% less data storage. Further, interpolation using VDVP-4X is approximately 56% more accurate than FDFP-4X for $N_C = 128$

Configuration		**Double Gyre**	Configuration		**ABC**	Configuration		**Tornado**
N_C	Data	Time Per Interval	N_C	Data	Time Per Interval	N_C	Data	Time Per Interval
2048	1X	**0.1704**	2048	1X	**0.4897**	128	1X	**3.5335**
	4X	**0.1774**		4X	**0.5669**		4X	**14.6271**
128	1X	**0.1821**	128	1X	**0.6528**	32	1X	**12.3912**
	4X	**0.1775**		4X	**1.0535**		4X	**50.4528**
8	1X	**0.2075**	8	1X	**4.1075**	8	1X	**54.5112**
	4X	**0.2712**		4X	**16.2275**		4X	**184.1937**

Table 1: *Timing results for post hoc interpolation using the VDVP-Interpolation method. For the Double Gyre data set, X = 262,144 points. For the ABC data set, X = 4.2M points. For the Tornado data set, X = 134.4M points. All timings reported are the average time for a single interval and measured in seconds.*

($P = 8192$), while using only 1.8% more data storage, and particle trajectories calculated using VDVP-2X are approximately 59% more accurate than FDFP-2X for $N_C = 32$ ($P = 16384$), while using 1.4% less data storage.

For the ABC data set, we observe the benefits of using VDVP-Interpolation given particles travel in relatively straight trajectories and maintain the same neighborhood for most interpolation steps (Section 7.2.2). Considering configurations between $P_{avg} = 1372$ to $P_{avg} = 125000$ in Figure 7b, VDVP-Interpolation is 6.5% more accurate while using 22% less data storage. Further, for multiple configurations our approach maintains accuracy while requiring less data storage. For example, interpolation using VDVP-2X has approximately the same accuracy as interpolation using FDFP-2X for $N_C = 2048$ ($P = 4096$), while using 20% lesser data storage, and interpolation using VDVP-4X is approximately 2.5% more accurate than using FDFP-4X for $N_C = 128$ ($P = 131072$), while using 30% less data storage.

For the Tornado plot in Figure 7c, considering all configurations, VDVP-Interpolation on average calculated particle trajectories that are 31% more accurate than the corresponding FDFP-Interpolation approach, while using 48% less data storage. Comparing VDVP-1X and FDFP-1X configurations, we observe that VDVP-Interpolation using 50% less data storage is approximately 60%, 47%, 40%, and 38% more accurate than corresponding FDFP-Interpolation accuracy for $N_C =$ 256, 128, 64, and 32 respectively. We placed seeds in areas from which particles are pulled into the vortex of the Tornado in the data set. Given the nature of the Tornado data set, we expect a lot of basis flows to exit the domain during the run and this contributes significantly to the lowered data storage. Overall, the interpolation accuracy of VDVP-Interpolation configurations is significantly better than the corresponding FDFP-Interpolation configurations across the board. We believe VDVP-Interpolation benefits from basis flows adapting and following the flow field thus offering better spatial resolution for the particles being interpolated.

7.2. VDVP-Interpolation Evaluation

In addition to accuracy and data storage we discuss the computation time required by our interpolation method. Further, to aid our understanding of the divergence in the flow field and the relation to interpolated pathline accuracy we observe the rate of neighborhood updates during interpolation.

7.2.1. Computation Time

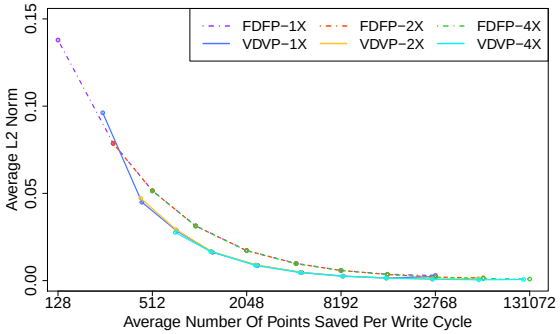
Table 1 shows the average time required for a single interval when performing VDVP-Interpolation for a select set of configurations. Given the number of intervals corresponds to the value of N_C, the total time required can be computed as a product of the average time per interval and N_C. Each interval of VDVP-Interpolation consists of identifying the next location for a set of particles and performing a serial Delaunay triangulation over the current set of input basis flows to identify the containing cell. Identification of the next location for each particle is computed using Barycentric coordinates interpolation and is computed in parallel over the total set of particles. For the configurations shown in Table 1, we believe the mid-value for N_C represents the most practical choice for configurations in practice.

For the Double Gyre data set, we observe short computation times given the 2D Delaunay triangulation is inexpensive for a relatively smaller number of points. Thus, the interpolation times are dominated by the parallel particle interpolation process. Overall, the total interpolation time is greater for high N_C and relatively low when N_C is small. For the ABC data set, we observe similar trends with total computation time proportional to the value of N_C. However, we do observe the impact of the 3D Delaunay triangulation over a large number of points as a bottleneck when N_C = 8 and VDVP-4X is used. For the Tornado data set, the 3D Delaunay triangulation dominates the total time required and is greater for VDVP-4X configurations. The number of intervals has a lesser effect when the Delaunay triangulation is expensive. For example, the total time required by the N_C = 128 and VDVP-1X configuration is less than the time required by the N_C = 8 and VDVP-4X configuration.

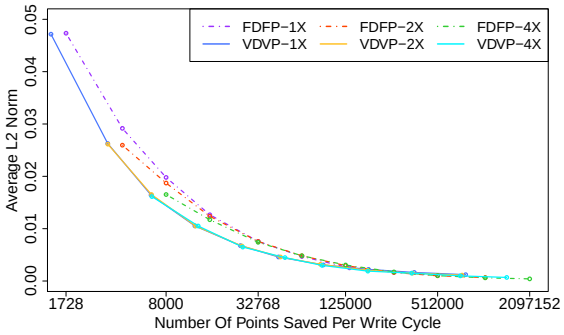
Our experiments with parallel calculation of the Delaunay triangulation showed it can significantly improve computation times, but there are constraints such as CGAL only offers a parallel 3D Delaunay triangulation which requires TBB (no support for 2D), and Delaunay triangulation on GPUs does not scale beyond a few million points due to memory constraints. We discuss potential solutions to this challenge in Section 9.

7.2.2. Neighorhood Update Rate

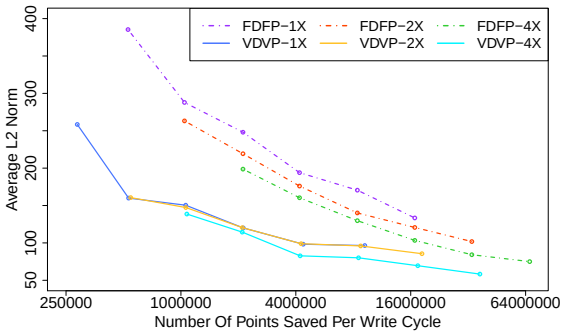
Figure 8 plots the average percentage neighborhood update rate of particles interpolated using the saved basis flow information. During VDVP-Interpolation, a particle follows a neighborhood of basis flows for an interval of time, followed by an evaluation of the



(a) *Double Gyre - L2-norm. X = 262,144 points.*



(b) *ABC - L2-norm. X = 4.2M points.*



(c) *Tornado - L2-norm. X = 134.4M points.*

Figure 7: *Evaluation results using L2-norm. Legend indicates the configuration information.*

continuation and quality of the neighborhood. If a member basis flow of the neighborhood terminates or the basis flows diverge beyond an acceptable threshold the particle identifies a new set of basis flows to follow by performing a neighborhood update. We use N_{update} to denote the average percentage neighborhood update rate. The FDFP-Interpolation approach has N_{update} equal to 100%.

For the Double Gyre data set, where we seeded particles randomly in the domain, we observe N_{update} is high when the intervals are large, i.e., the number of write cycles is small. This is expected given the circulating and diverging nature of the Double Gyre flow. For the ABC data set, where we seeded particles randomly in the domain, we observe low values for N_{update} irrespective of the number of write cycles. This is expected given particles have rather

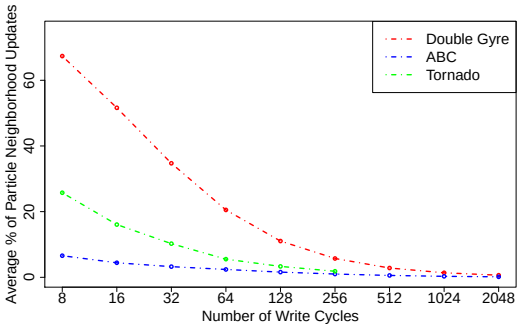


Figure 8: *The average percentage neighborhood update rate over all particles for each data set. The x-axis shows the number of write cycles, i.e., the number of "opportunities" a particle being interpolated had to evaluate and decide whether to update its neighborhood. The y-axis plots the average percentage of neighborhood updates over all interpolated particle trajectories, considering all configurations, i.e., VDVP-1X, VDVP-2X, and VDVP-4X.*

straight trajectories for this data set, thus being able to maintain the same neighborhood. For the Tornado data set, we seeded particles at select locations in order to capture the flow of the mature vortex in the field. We observe that for the large intervals, particles have N_{update} approximately equal to 25%. For all data sets, for small temporal intervals (i.e., high values of N_C), we observed low values for N_{update}. Thus, particles choose to continue using the same neighborhood of basis flows for upcoming interpolations.

8. Conclusion

Our interpolation scheme VDVP-Interpolation reduces error propagation and limits interpolation error when calculating particle trajectories using VDVP Lagrangian basis flows. VDVP-Interpolation makes configurable neighborhood-aware usage of VDVP basis flows and is the main contribution of this work. Further, our work is the first practical implementation of generating and using VDVP basis flows. This serves as a starting point for future in situ methods research for flow analysis and visualization using the Lagrangian paradigm.

We evaluated the accuracy-storage propositions offered by our method for multiple data sets and demonstrate improved accuracy and reduced storage compared to previous methods. For example, VDVP-Interpolation was able to calculate particle trajectories that were between 40%-60% more accurate while using 50% less storage for certain configurations of the Tornado data set.

9. Future Work

Identifying the neighborhood of a particle can be expensive using a global search structure which requires construction over a large number of points or updating every interval. An alternative approach could be a local parallel search for nearby, relevant basis flows to form the neighborhood for each particle. Identifying an efficient approach for tracking particle neighborhoods for unstructured input will be explored as a future research direction with options including spatial hashing and binning being considered.

Significant research needs to be directed toward the best approaches for generating VDVP basis flows. Generating VDVP basis flows in an in situ distributed environment introduces new challenges with regard to particle distribution management, communication costs, and scalability given the complexity of distributed time-varying integral curve computation. While the Lagrangian paradigm offers significant flexibility in terms of flow sampling, efficient in situ approaches to capture interesting regions of the flow need to be researched and developed.

Acknowledgment

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

[ACG*14] AGRANOVSKY A., CAMP D., GARTH C., BETHEL E. W., JOY K. I., CHILDS H.: Improved post hoc flow analysis via lagrangian representations. In *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on* (2014), IEEE, pp. 67–75. 1, 2

[AGJ11] AGRANOVSKY A., GARTH C., JOY K. I.: Extracting flow structures using sparse particles. In *VMV* (2011), pp. 153–160. 2

[ASM*11] AHERN S., SHOSHANI A., MA K.-L., CHOUDHARY A., CRITCHLOW T., KLASKY S., PASCUCCI V., AHRENS J., BETHEL W., CHILDS H., ET AL.: Scientific discovery at the exascale: report from the doe ascr 2011 workshop on exascale data management, analysis, and visualization. *Dept. of Energy Office of Advanced Scientific Computing Research* (2011). 1

[BAA*16] BAUER A. C., ABBASI H., AHRENS J., CHILDS H., GEVECI B., KLASKY S., MORELAND K., O'LEARY P., VISHWANATH V., WHITLOCK B., ET AL.: In situ methods, infrastructures, and applications on high performance computing platforms. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 577–597. 2

[BCT01] BRUMMELL N., CATTANEO F., TOBIAS S.: Linear and nonlinear dynamo properties of time-dependent abc flows. *Fluid Dynamics Research 28*, 4 (2001), 237–265. 6

[BJ15] BUJACK R., JOY K. I.: Lagrangian representations of flow fields with parameter curves. In *Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on* (2015), IEEE, pp. 41–48. 1, 2, 3, 7

[CBJ16] CHANDLER J., BUJACK R., JOY K. I.: Analysis of error in interpolation-based pathline tracing. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers* (2016), Eurographics Association, pp. 1–5. 2, 4

[CK90] CASH J. R., KARP A. H.: A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS) 16*, 3 (1990), 201–222. 7

[COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE transactions on visualization and computer graphics 21*, 1 (2015), 68–80. 2

[GM77] GINGOLD R. A., MONAGHAN J. J.: Smoothed particle hydrodynamics-theory and application to non-spherical stars. *Monthly notices of the royal astronomical society 181* (1977), 375–389. 2

[HBJG16] HUMMEL M., BUJACK R., JOY K. I., GARTH C.: Error estimates for lagrangian flow field representations. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers* (2016), Eurographics Association, pp. 7–11. 1, 2, 3

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE transactions on visualization and computer graphics 17*, 8 (2011), 1148–1163. 2

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing?97*. Springer, 1997, pp. 43–55. 2

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum 23* (2004), 2004. 2

[LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 965–972. 2

[MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest point seeding for efficient placement of streamlines. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 479–486. 2

[MLP*09] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over Two Decades of Integration-Based, Geometric Flow Visualization. In *EG 2009 - State of the Art Reports* (2009), pp. 73–92. 2

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1807–1829. 2

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics* (2003), ACM, pp. 213–222. 2

[OWW15] ORF L., WILHELMSON R., WICKER L.: Visualization of a simulated Long-Track EF5 tornado embedded within a supercell thunderstorm. *Parallel Comput. 0*, 0 (2015). in press. 6

[PPF*11] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIC K., HAUSER H.: The State of the Art in Topology-based Visualization of Unsteady Flow. *Computer Graphics Forum 30*, 6 (September 2011), 1789–1811. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.01901.x. 2

[RPP*09] ROSANWO O., PETZ C., PROHASKA S., HEGE H.-C., HOTZ I.: Dual streamline seeding. In *Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific* (2009), IEEE, pp. 9–16. 2

[SBC18] SANE S., BUJACK R., CHILDS H.: Revisiting the Evaluation of In Situ Lagrangian Analysis. In *Eurographics Symposium on Parallel Graphics and Visualization* (2018), Childs H., Cucchietti F., (Eds.), The Eurographics Association. doi:10.2312/pgv.20181096. 2, 3

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3 (2005), 271–304. 4, 6

[SXM16] SAUER F., XIE J., MA K.-L.: A combined eulerian-lagrangian data representation for large-scale applications. *IEEE Transactions on Visualization and Computer Graphics* (2016). 2

[VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *Proceedings of the conference on Visualization'00* (2000), IEEE Computer Society Press, pp. 163–170. 2

[Wal98] WALDRON S.: The error in linear interpolation at the vertices of a simplex. *SIAM Journal on Numerical Analysis 35*, 3 (1998), 1191–1200. 5

[WLZMI10] WU K., LIU Z., ZHANG S., MOORHEAD II R. J.: Topology-aware evenly spaced streamline placement. *IEEE Transactions on Visualization and Computer Graphics 16*, 5 (2010), 791–801. 2

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1216–1224. 2

[ZSH96] ZOCKLER M., STALLING D., HEGE H.-C.: Interactive visualization of 3d-vector fields using illuminated stream lines. In *Visualization'96. Proceedings.* (1996), IEEE, pp. 107–113. 2