Volume xx (200y), Number z, pp. 1–12

Improving Flow Analysis Using In Situ Lagrangian Techniques

Directed Research Project Report

Sudhanshu Sane
Advisor: Hank Childs

Abstract

We introduce a new approach, consisting of two complimentary techniques, to perform in situ Lagrangian flow analysis. Previous work had placed constraints on the placement and duration of basis flows, namely putting them in regular positions, and forcing all basis flows to terminate at the same time. With our work, we relax these constraints in order to achieve better accuracy. Our first technique guides the creation of basis flows. It differs from previous work in that we allow the duration of basis flows to vary. The varying duration allows for us to create long-lived basis flows, which improves accuracy by preventing error propagation when interpolating new particle trajectores during post hoc analysis. Our technique also differs from previous work in that we identify regions of greatest need, via Delaunay triangulation, and then place basis flow seeds in these regions. Our second technique is for interpolation of basis flows of arbitrary seed placement and durations: this technique is necessary since our first technique generates such data and since there is no previous work on interpolation of basis flows with arbitrary durations. We show that our complementary techniques produce improved accuracy and demonstrate the potential for reduced data storage. This means we can provide improved accuracy with same or less storage, or same accuracy with less storage.

1. Introduction

1.1. The Two Flow Specifications

In fluid dynamics, two frames of references are typically considered: Eulerian and Lagrangian. The Eulerian frame of reference describes the flow with respect to a fixed position and time. The Lagrangian frame of reference describes a flow parcel as it moves through space and time in the flow field.

Traditionally, the Eulerian specification is used in flow visualization. It is usually stored in the form of a series of vector fields, each representing one time slice. Many visualization techniques, such as pathlines, path surfaces, and FTLE, require calculation of particle trajectories. These trajectories can be calculated from the vector field information through numerical integration, which requires temporal interpolation between the stored time slices.

The Lagrangian specification stores information in the form of a flow map. The flow map $F_{t_0}^{t}(x_0) : R^d \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^d$ 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 traditional approach, the stored information represents a time interval and new particle trajectories can be computed simply through interpolation using the basis of known trajectories.

1.2. The I/O Bottleneck and In Situ Processsing

Fluid dynamics experiments are regularly simulated on high performance supercomputers. To simulate the flow field accurately, the scientific simulations are advanced in small discrete time steps. Each of these time steps is called a cycle. Modern supercomputer architectures demonstrate ever increasing computational capabilities. However, I/O capabilities on these supercomputers have not kept pace with their ability to generate data. Given the growing gap between computational and I/O capability, the frequency of storing flow field information is reducing. Under this temporal sparsity, post hoc flow analysis using the Eulerian specification can be adversely affected.

An emerging paradigm to counteract this temporal sparsity is the use of in situ processing. 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 is capable of representing more information per byte stored.

1.3. State of In Situ Lagrangian Techniques

Agranovky et al. [ACG*14] presented a two stage process to perform flow analysis using a Lagrangian-based approach. The first stage involves extracting trajectories during the simulation (in situ). These trajectories are called basis flows. The second stage involves

interpolating new trajectories from the basis flows after the simulation (post hoc). In effect, the behavior of the flow field is approximated using the basis flows. This paradigm 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.

For the in situ phase of Agranovsky et al.'s work, the basis flows are calculated in batches. 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, introducing new particles again along a uniform grid and terminating them after a fixed number of cycles, until the simulation completes.

For the post hoc phase of Agranovsky et al.'s work, the basis flows from the in situ 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. Interpolation of 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 next batch of basis flows. The process is repeated until the particle reaches its desired termination time. Agranovsky et al.'s study showed that using the Lagrangian approach is superior to the Eulerian approach under sparse temporal settings.

For the remainder of the paper, we refer to the described Agranovsky et al. method as Uniform Seeding.

1.4. Our contribution

The Uniform Seeding approach has two constraints when calculating the basis of known trajectories. First, the placement of particles to calculate basis flows is always along a uniform grid. Second, the duration of each basis trajectory is fixed, resulting in a series of short basis flows. The primary drawback of these constraints is that it results in an approach that is prone to error propagation during the post hoc calculation of trajectories [BJ15].

Our work aims to improve accuracy by reducing the occurrence of error propagation events. A particle can be interpolated more accurately when it is following the same neighborhood of basis flows for a longer duration. With the goal of achieving improved flow analysis, we relax the constraints of Uniform Seeding. Specifically, our approach employs variable duration basis flows, including the dynamic addition of helpful basis flows and termination of unneeded basis flows. Further, the new basis flows are added intelligently and do not need to lie along uniform locations.

All in all, with this work we present an end-to-end in situ Lagrangian technique which minimizes error propagation and accumulation, resulting in improved accuracy of post hoc flow analysis and visualization. Considered separately, the main contributions of our work are as follows:

- An in situ seed placement strategy that is tailored towards minimizing error propagation.
- A corresponding post hoc interpolation scheme to make optimal usage of information generated in situ.

The rest of this paper is organized as follows. Section 2 covers work related to Lagrangian techniques and seed placement strategies for flow analysis. Section 3 motivates the issues we address with our work, with Section 4 containing our proposed algorithm details. The details of our study and our findings are presented in Sections 5 and 6, respectively.

2. Related Work

2.1. Flow Analysis using Lagrangian Techniques

Particle trajectories are one of the fundamental elements of flow visualization [LHD*04, MLP*09, PPF*11]. They can be used directly as pathines to represent the behavior of flow in their vicinity, or in more advanced ways like path surfaces [MLZ09], classifying regions of different behavior [SGSM08,STH*09,HKTH16], for topological analysis [TWHS05, SW10], and as the basis for the finite time Lyapunov exponent [HY00, GGTH07, GLT*09, KHH12].

In addition to error propagation issues due to the I/O constraints described in Section 1, the abundance of Lagrangian-based visualization techniques motivated the use of a flow map over a vector field to represent a flow. In their work, Agranovsky et al. [ACG*14] introduced this technique in the context of reducing storage and error for in situ supercomputing environments. Earlier, Hlawatsch et al. [HSW11] stored flow field data directly. They focused on precomputing Lagrangian-based trajectories and optimally selecting which to use when calculating pathlines. They employed a hierarchical scheme to decrease the number of integration steps by constructing longer integral lines from previously computed partial solutions. Agranovsky et al. [AGJ11] studied the use of Moving Least Squares, barycentric coordinate interpolation, to optimize pathline interpolation using scattered particles.

The traditional approach for computational fluid dynamics simulations is based on finite elements and producing vector data. An alternative method is Smooth Particle Hydrodynamics (SPH) [GM77]. This method's natural output is particle trajectories, which makes it well suited for Lagrangian techniques. Chandler et al. [COJ15] proposed a modified k-d tree to store particle locations at a given time and an interpolation scheme that is tailored towards the kernel native to the simulation. Sauer et al. [SXM16] presented a new data representation which combines the Eulerian and Lagrangian reference frames into a joint format.

Research has also been focused on identifying sources of error in advection methods used in the Lagrangian paradigm. Chandler et al. [CBJ16] presented an error analysis of their interpolation based pathline tracing system and experimentally found that error roughly correlates with divergence in flow fields. Bujack et al. [BJ15] derived theoretical error bounds for the Lagrangian method and identified that the source of the error propagation is neighborhood updates. Further, they suggested using parameter curves, such as Bezier curves and cubic Hermite splines instead of polygonal chains, to improve the aesthetics of flow visualizations. Hummel et al. [HBJG16] extended this work by using upper error bounds to visualize the uncertainty of the pathlines.

2.2. Seed Placement Techniques for Flow Analysis and Visualization

Prior works related to seed or streamline placements for flow visualization have had objectives such as coverage of interesting regions in the vector field, uniformity in the distribution of visualized streamlines over the field, and aesthetics of the visualization [VKP00]. While the visualization of steady state flow fields through intelligent placement of seeds and their resulting streamlines has been extensively researched, seed placement strategies for unsteady state flow fields has been less explored [MLP*10]. Visualizations of unsteady state flow fields is often done by the animation of streamlines [JL00].

Turk and Banks [TB96] introduced an image guided streamline placement algorithm to achieve a uniformly dense streamline coverage. Jobard and Lefer [JL97] worked on a strategy which was computationally more efficient and aimed to control the density of streamlines via user defined parameters for distance between streamlines. Mattausch et al. [MTHG03] extended this idea to 3D. Verma et al. [VKP00] presented a method of streamline placement that captures the flow patterns around critical points. They used a set of predefined templates to handle each case of critical points in 2D vector fields. After that, population of sparse regions of the flow field was achieved by a Poisson disk distribution. This concept was extended to 3D by Ye et al. [YKP05]. Mebarki et al. [MAD05] adopted an approach, which used Delaunay triangulation to identify cavities in the field and then placed seeds at the centroid of the triangle. This was followed by forward and backward integration to draw a streamline. This process was repeated until a desired density threshold is met. The approach ensured that visually appealing long streamlines were produced. Liu et al. [LMG06] built on previous works to present another evenly spaced streamline algorithm. They used cubic Hermite polynomial interpolation to create fewer evenly spaced streamline samples in the neighborhood of each previous streamline, thus reducing the amount of distance checking. Zockler et al. [ZSH96] used scalar values, such as velocity magnitude, to determine the degree of interest of a cell. An equalization strategy was then used to distribute seed points more homogeneously. An image-based streamline placement strategy for 3D vector fields was proposed by Li et al. [LS07].

Similarity measures have been used to avoid dense sets of streamlines or to group similar streamlines together to capture important flow field features using a minimal representation [YWSC12, CCK07, LHS08, MJL*13]. Marchesin et al. [MCHM10] introduced a view-dependent streamline placement and selection algorithm for 3D vector fields which aimed at avoiding occlusion. Wu et al. [WLZMI10] proposed a topology-aware evenly spaced streamline placement algorithm for capturing singularities and separatrices.

For 3D unsteady state flows, Wiebel and Scheuermann [WS05] focused on regions of high activity and interest. They presented a method involving bundles of streaklines and pathlines passing through a single point in space, referred to as an eyelet, at different times. Helgeland and Elbroth [HE06] focused on the visualization of the 3D unsteady flow field while adopting a seeding algorithm based on Jobard and Lefer's even spaced streamline seeding algorithm. Obermaieret et al. [OHBKH09] implemented distance- and angle-based reseeding and removing strategies for improving streakline- and timeline-based surfaces in gridless flows.

Seed placement strategies have been researched for improving the accuracy of FTLE, since frequent renormalization becomes necessary in highly divergent time-varying flow fields [Nes89, HY00]. Two separate works, Garth et al. [GGTH07] and Sadlo et al. [SP07], suggested adaptive reseeding methods that improved the precision of the gradient of the flow map. Kuhn et al. [KRWT12] compared different seed placement methods with respect to the accuracy of FTLE. These works aimed to optimize the computation of the gradient and are mainly restricted to orthogonal meshes.

With our own work, we look at seed placement strategies for calculating basis flows, with the objective of accurate post hoc flow field reconstruction, i.e., minimizing the linear error of barycentric interpolation. Further, our technique is developed exclusively for unsteady state flow fields. Agranovsky et al. [ACJC15] used flow field information such as turbulence and vorticity in situ to make an informed selection of vector field sample locations for Eulerian post hoc particle advection. With regard to particle placement strategies using the Lagrangian paradigm, as mentioned in Section 1.3, Agranovsky et al. [ACG*14] placed seeds along a uniform grid periodically to calculate basis flows.

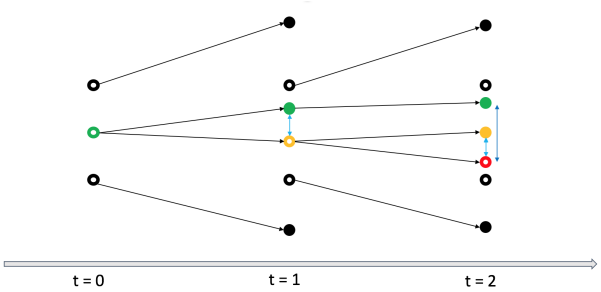
3. Motivation

Problem: The Uniform Seeding approach suffers from error propagation. As described in Section 1.3, Agranovsky et al.'s approach [ACG*14] is a numerical one-step integration method, which suffers from error propagation. A particle is advanced in time by following a neighborhood of basis flows. However, given that the basis flows are calculated in batches for the Uniform Seeding approach, the process requires identification of a new neighborhood, i.e., a neighborbohood 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 lead to poor accuracy.

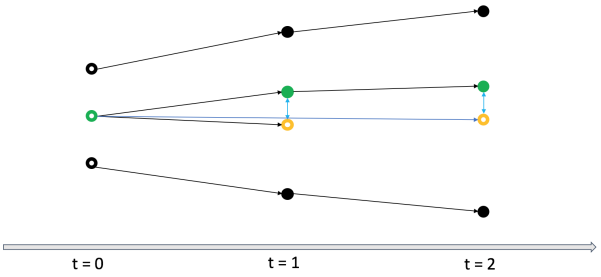
Proposed Solution: Extend the duration of basis flows for as long as possible. The error propagation and accumulation occurs for every instance of a stitching event (neighborhood update). We can mitigate this issue if:

1. Basis flows live for the duration of the simulation.
2. The interpolation is done based on the initial neighborhood information. This is possible because having the basis flows live for the duration of the simulation means a particle has 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



(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 incorrect position 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 holllow red) is $(\frac{1}{2}h_x^2\|f''\|)$. The local error propagates with each interpolation. The global error is enhanced by the Lipschitz constant h_tL of f. Thus, at $t = 2$, the global error is already $\frac{1}{2}h_x^2\|f''\|(1+h_tL)$ [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 the advantage of using longer basis flows to reduce error propagation.

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. Figure 1b illustrates a particle being interpolated by using the same neighborhood.

The Uniform Seeding approach is a numerical one-step integration method, while the proposed approach uses purely interpolation. To highlight the difference in error propagation and accumulation between the two approaches, we consider an analytic flow field — a distorted circular flow. Figure 2 shows that the pure interpolation approach 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. 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, 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 Uniform Seeding). 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 under represented regions could be poor.

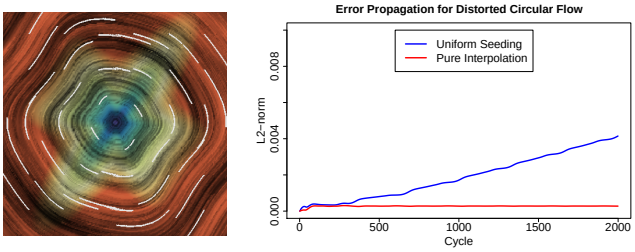
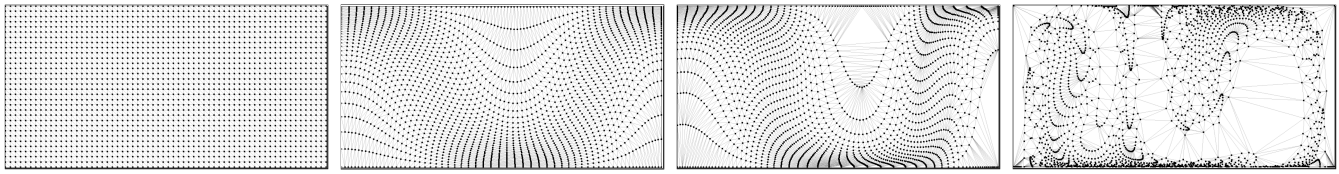


Figure 2: We used a test analytic data set — a distorted circular flow. The image on the left is the line integral convolution of the flow field. The color encodes the velocity magnitude. The white lines are the Uniform Seeding basis trajectories. The image on the right is a plot of error propagation over 2000 cycles. In contrast to Uniform Seeding, the pure interpolation has absolutely no error propagation.

Proposed Solution: Extend the duration of basis flows for as long as possible, but update the particle neighborhood if it exceeds a limit for h_x. In this paper, we propose a hybrid approach between the uniform case and the pure interpolation approach. Thus, when performing post hoc interpolation, as long as a particle lives in a non-divergent neighborhood, it uses the pure interpolation approach. As soon as a particle's neighborhood diverges, the neighborhood of the particle is updated. In order to guarantee that a small neighborhood can always be found, we evaluate the flow field in situ to determine when new particles should be introduced to prevent poor coverage of regions in the field. By introducing new particles intelligently, we can guarantee a bounded interpolation error while minimizing error propagation and accumulation events, i.e., particle neighborhood updates. The following section details our proposed algorithm which includes an in situ basis flow seed placement strategy and a post hoc interpolation scheme.

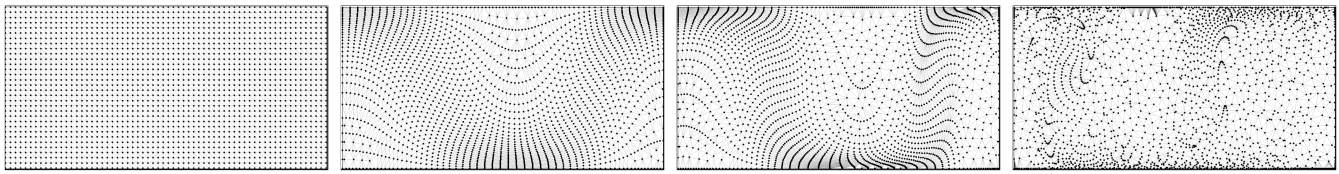
4. Lagrangian Flow Seeding Method

Our proposed in situ Lagrangian technique follows the same high level approach as described in Section 1.3. It is a two stage process. In the first stage, basis flows are calculated in situ and saved to disk. Our solution is different from previous work in that it focuses on longer basis flows (to prevent error propagation), as well



(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.



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

Figure 4: Flow Seeding particle distribution for the Double Gyre, with period set to 1000 cycles. Many neighborhoods stay together over the whole period and allow advection free of error propagation.

as addressing a problem that comes from longer basis flows, which is the presence of under represented regions. In the second stage, the basis flows are used to perform flow field analysis post hoc. Our solution is again different, since we needed to design a viable interpolation technique that can effectively use the basis flows from our first stage.

We refer to our proposed technique as Flow Seeding.

4.1. In Situ Seed Placement Strategy

When generating basis flows in situ, we have control over where we want to place our particles. Additionally, we can choose how long we allow a particle to exist. Our primary objective is to generate basis flows which enable accurate post hoc flow analysis. With the Flow Seeding 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.

To perform particle addition and removal, we consider all existing particles in the volume at the end of a write cycle. Let the set of existing particle locations be **S**. We perform a Delaunay Triangulation on **S** resulting in triangulation **DT**. Thus, each vertex in **DT** represents a particle location. **DT** is a set of cells that are triangles or tetrahedrons depending on whether the dataset is 2D or 3D, respectively. By definition, the interior of each cell defined by the Delaunay triangulation, contains no vertices., i.e., no particles. We then modify **DT** to add and remove vertices (see Sections 4.1.1 and 4.1.2); the remaining vertices represent the particles that are valid and can either continue as basis flows or form new basis flows. Figure 4 shows the distribution of particles, at various stages of the Double Gyre, when using Flow Seeding.

4.1.1. Particle Addition

Particle addition means that new basis flows are introduced when flows diverge. To identify large cells in the triangulation **DT** we measure the circumradius of each cell in **DT**. Let R denote the circumradius of a cell. Cells with large circumradius R represent candidate regions for particle addition.

We use barycentric coordinate interpolation post hoc to interpolate the trajectory of a particle. 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].

Thus, a large circumradius corresponds to a possibly large post hoc interpolation error while using the particles forming that cell. We iterate over all cells of **DT** to determine the cell with the largest circumradius. If the largest circumradius is greater than a predetermined threshold $UpperThreshold$, we insert a vertex into **DT** as the seed location of the new particle. The seed location is determined as follows:

- If the circumcenter is located inside the cell, then the seed location is at the circumcenter of the cell.
- If the circumcenter is located outside the cell, then the seed location is the point on the boundary of the cell that is closest to the circumcenter.

A local Delaunay triangulation insert is performed on **DT** to include the new particle before further iterations. This process is continued until there are no cells in **DT** with a circumradius greater than the threshold. Algorithm 1 shows the steps involved for particle addition.

```
Function ParticleAddition(DT, UpperThreshold)
AddParticles = true;
MaxCircumradius = 0;
MaxCell;
while AddParticles do
    foreach Cell c ∈ DT do
        if c.circumradius > MaxCircumradius then
            MaxCell = c;
            MaxCircumradius = c.circumradius;
        end
    end
    if MaxCell.circumradius > UpperThreshold then
        Seed = calculateLocationOfSeed(MaxCell);
        DT.insertVertex(Seed);
    else
        AddParticles = false;
    end
end
```

Algorithm 1: Particle Addition Algorithm

4.1.2. Particle Removal

For particle removal, we determine if a particle must be removed by iterating over all vertices of **DT** and calculating the average of the circumradii of all the cells that the vertex, representing the particle, is a member of. If the lowest calculated average circumradius is below a predetermined threshold $LowerThreshold$, the associated vertex is removed from **DT**. This process is continued until there are no vertices with an average circumradius lower than the threshold. Particle removal results in the basis flow ending. Algorithm 2 shows the steps involved for particle removal.

```
Function ParticleRemoval(DT, LowerThreshold)
RemoveParticles = true;
MinVertex;
MinAvgCircumradius = inf;
while RemoveParticles do
    foreach Vertex v ∈ DT do
        Cells = DT.getCellsContainingVertex(v);
        SumCircumradius = 0;
        foreach Cell c ∈ Cells do
            SumCircumradius += c.circumradius;
        end
        AvgCircumradius = SumCircumradius/Cells.size();
        if AvgCircumradius < MinAvgCircumradius then
            MinVertex = v;
            MinAvgCircumradius = AvgCircumradius;
        end
    end
    if MinAvgCircumradius < LowerThreshold then
        DT.deleteVertex(MinVertex);
    else
        RemoveParticle = false;
    end
end
```

Algorithm 2: Particle Removal Algorithm

4.2. Post Hoc Interpolation Scheme

We propose a new interpolation scheme that can make use of variable duration basis flows. 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. For a given particle location P_0 at time T_0, our interpolation scheme starts by identifying a neighborhood of basis flows $B_1, B_2, \ldots, B_n$ 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. 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.

To identify a particle neighborhood at time T_i, we first perform a Delaunay triangulation over basis flow particle locations at time T_i. A particle neighborhood is then 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, $UpperThreshold$.

Using the proposed interpolation scheme, a particle can follow the same neighborhood of basis flows for a longer duration if they meet certain criteria. A neighborhood is only updated if a member basis flow is terminated or the cell formed by the neighborhood has a circumradius exceeding $UpperThreshold$. Additionally, we use $UpperThreshold$ when determining whether to add a new particle during the in situ phase of Flow Seeding, meaning we guarantee a neighborhood with circumradius below $UpperThreshold$ can always be found.

5. Study Overview

We designed a study to evaluate our Flow Seeding approach. Further, we performed the same experiments using the Uniform Seeding approach and use the results as a baseline for comparison. For

our study, the in situ environment was theoretical, evaluating analytic data sets on the fly or loading simulation results that were precalculated for each cycle from disk. Our study consists of configurations which vary over five parameters:

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

We ran a total of 192 test configurations. We performed all the tests on a Xeon E5-2667v3 CPU. We used 12 cores at 3.2GHz and 256 GB DDR4 memory. In situ basis flow calculation and post hoc particle trajectory calculation were performed in parallel. To we used the CGAL library to serially calculate the Delaunay triangulation, and to perform vertex insertion and deletion.

5.1. Configuration Parameters

5.1.1. Lagrangian techniques

A test is configured to use either Flow Seeding (our technique) or Uniform Seeding (the technique from Agranovsky et al.) as its in situ Lagrangian technique. To allow for comparison, we ran each test configuration with each technique.

5.1.2. 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. 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. We set the period of the Double Gyre flow to 1000 cycles.

Arnold-Beltrami-Childress (ABC) — This data set is a time-dependent variant of the three-dimensional ABC analytic vector field [BCT01]. The data set is simulated for 2048 cycles at a base resolution of $128 \times 128 \times 128$. We set the period of the ABC flow to 1000 cycles.

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 vertex 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" for his studies. Thus, we considered 257 time slices, with the time-steps evenly distributed from $t_0 = 8502$s to $t_{256} = 9014$s.

5.1.3. Data Storage Targets, Number of Cycles Saved, and Number of Basis Flows Saved per Cycle

With the Uniform Seeding approach, the number of basis flows saved every write cycle can be fixed. However, for the Flow Seeding method, the number of basis flows should be viewed as a target, as the number of basis flows fluctuates over time. First, basis flow particles are terminated if the trajectory of the particle exits the domain space. Further, basis flows are added or removed based on user-defined thresholds.

Thus, for each test configuration we give the Uniform Seeding approach a fixed data storage target for the entire simulation. Let P denote the number of basis flow particles stored during a write cycle. Let N_C denote the number of cycles saved, i.e., the number of write cycles. Then, if X denotes the data storage target, we select combinations of P and N_C such that $P \times N_C = X$.

We give the Flow Seeding approach the same number of initial basis flow particles P as the Uniform Seeding approach. Thus, both approaches begin with the same number of basis flow seed particles. We can then evaluate the performance of the Flow Seeding approach with regard to data storage costs.

For our study, we consider four data storage targets for each data set. We believe the results should be representative for other data storage targets. For each data storage target of each data set, we select multiple configurations that are combinations of P and N_C. The Double Gyre and ABC data sets each have nine configurations for a given target, while the Tornado data set has six configurations for each target.

Let the targets for each data set be represented as $1X$, $2X$, $4X$ and $8X$, with X varying depending on the data set. For the Double Gyre data set, $X = 512 \times 256 = 131,072$ points. For the ABC data set, $X = 128 \times 128 \times 128 = 2.1$M points. For the Tornado data set, $X = 490 \times 490 \times 280 = 67.2$M points.

The Uniform Seeding approach will use 100% of the data storage target each time, while the Flow Seeding approach data storage costs are expected to vary. As previously mentioned, particle addition and removal influences the total data storage costs. 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 use $C = 2$ for the two-dimensional Double Gyre data set, and $C = 8$ for the three-dimensional ABC and Tornado data sets.

5.2. Error Evaluation

We calculate particle trajectory using three different methods.

- **Ground Truth** — The particle trajectory is calculated with a fourth-order Runge Kutta scheme [CK90]. The ground truth uses the full spatial and temporal resolution available to calculate the trajectory. The ground truth is considered to be perfectly accurate, i.e., it has 0% error.
- **Uniform Seeding** — These Lagrangian trajectories are interpolated using second order Barycentric coordinate interpolation for each configuration of N_C and P. In this case, P is the number of basis flow particles placed along a uniform grid in the volume initially and at the end of each write cycle.
- **Flow Seeding** — These Lagrangian trajectories are interpolated using second order Barycentric coordinate interpolation for each

configuration of N_C and P. In this case, P is only the initial number of basis flow particles seeded in the volume.

Every test configuration generates a set of trajectories using either the Uniform or Flow Seeding approach. We evaluate the accuracy of 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. We then calculate the set of trajectories for each test configuration.

To compare two trajectories we measure the L2-norm. As previously defined, N_C is the number of cycles saved. Furthermore, N_C is the number of particle positions used to represent a Lagrangian trajectory.

The average L2-norm is calculated as follows —

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

where p is the total number of particles, n is equal to N_C, $x_{i,t}$ 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 known points of the Lagrangian trajectory can be connected using linear interpolation or curve fitting. Representation of a Lagrangian trajectory as a whole has been studied by Bujack et al. [BJ15]. For our study, we focus on the accuracy of the interpolated locations of a particle trajectory.

6. Results

We evaluated the Flow Seeding approach on three axes: data storage costs, interpolated trajectories accuracy, and computation time of each phase.

6.1. Data Storage

To evaluate the data storage costs of the Flow Seeding approach, we account for all the basis flow information saved during the in situ phase. We compare these results to the data storage targets set. We used the data storage targets to determine the test configurations of both the Flow Seeding and Uniform Seeding approach. Table 1 shows the data storage costs of the Flow Seeding approach in terms of average percentage usage of the data storage target. Data storage costs for the Flow Seeding approach depend on the nature of the flow field and is influenced by the particle addition and removal thresholds: $UpperThreshold$ and $LowerThreshold$ respectively. The Uniform Seeding approach always uses 100% of the data storage target.

Double Gyre — The first row in Table 1 shows the data storage costs for the Double Gyre data set. On average, the Flow Seeding approach uses more data storage than the set target. For our study, for a constant data storage target, N_C and P are inversely proportional to each other, We observed that data storage cost is high when N_C is high and P is low, and the data storage cost is low when N_C is low and P is high. The overall high data storage costs are largely attributed to the nature of the flow field: no particles exit the boundaries of the flow field. However, we do observe that as the data storage target increases, the Flow Seeding approach gets closer to achieving the target. Thus, given a larger data storage allowance for the Double Gyre, we expect the Flow Seeding approach to use the same or lesser storage than Uniform Seeding.

Data Set	Data Storage Costs				
Double Gyre	Target	1X	2X	4X	8X
	% Usage	144%	124%	113%	106%
ABC	Target	1X	2X	4X	8X
	% Usage	86%	80%	77%	75%
Tornado	Target	1X	2X	4X	8X
	% Usage	53%	50%	50%	49%

Table 1: Average percentage usage of the data storage target by Flow Seeding. Uniform Seeding uses 100% of the data storage target.

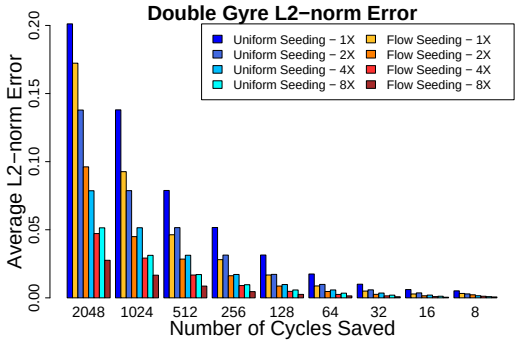
ABC — The second row of Table 1 shows the data storage costs for the ABC data set. Given the nature of the ABC field, initially placed basis flow particles will eventually exit the volume. This characteristic of the ABC flow contributes to the reduced data storage costs while using the Flow Seeding approach. We observe that for every data storage target considered, the Flow Seeding approach uses lesser storage on average than the set target. Once again, the percentage usage decreases as the overall data storage target increases. On average, for all data storage targets considered, it uses approximately 20% less storage than Uniform Seeding.

Tornado — The third row of Table 1 shows the data storage costs for the Tornado data set. The Flow Seeding approach uses less storage than every considered data storage target. On average, for all data storage targets considered, it uses approximately 50% less storage than Uniform Seeding. Similar to the ABC field, basis flow particles exit the volume and our in situ placement technique adds particles to prevent poor coverage of the field.

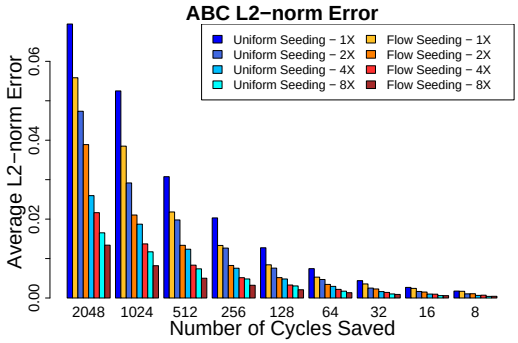
Overall, we observed potential for improved data storage costs with the Flow Seeding approach, especially when considering higher data storage allowance. Further, we believe there are several ways to control data storage costs, such as: placing an absolute upper bound on particles addition, enforcing an equal number of additions and removals, adjusting $UpperThreshold$ and $LowerThreshold$ to influence particle addition and removal, and so on. With our work, we chose to place an upper bound on post hoc interpolation error.

6.2. Accuracy

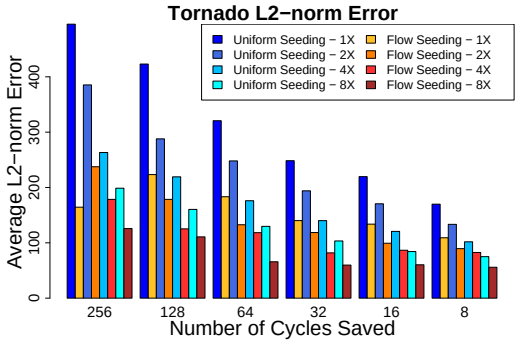
As mentioned in Section 5.2, our accuracy measurements use the L2-norm. Figure 5 plots accuracy measurements for all the data sets considered. Overall, for all data sets, we observe trends consistent across the parameter space. Flow Seeding has lower error than every corresponding Uniform Seeding result. Further, for high N_C configurations, we observe the Flow Seeding approach performs significantly better than the Uniform approach. In particular, these configurations demonstrate the improved accuracy from reduced error propagation.



(a) Double Gyre - L2-norm. X = 131,072 points.



(b) ABC - L2-norm. X = 2.1M points.



(c) Tornado - L2-norm. X = 67.2M points.

Figure 5: Evaluation results using L2-norm. Legends indicate the total data storage target information.

Double Gyre — When observing the accuracy measurements shown in Figure 5a, we must take the data storage costs into consideration. We find several instances where accuracy of Flow Seeding is better than Uniform Seeding while using less data storage. For example, Flow Seeding initialized with 4X data storage target is on average approximately 10% more accurate than Uniform Seeding using 8X data storage. Similarly, Flow Seeding initialized with a 2X data storage target, achieves better accuracy (approximately 5% more accurate on average) than Uniform Seeding using 4X data storage for several configurations. Flow Seeding initialized with 8X data storage target only used 6% more basis flows on average than its corresponding Uniform Seeding configuration, while producing Lagrangian trajectories that are approximately 52% more accurate.

ABC — We notice several instances of Flow Seeding achieving accuracy similar to Uniform Seeding, while using half the data storage. For example, when considering $N_C = 256$ in Figure 5b, we observe Flow Seeding 1X has error similar to Uniform Seeding 2X, Flow Seeding 2X has error similar to Uniform Seeding 4X, and so on. On average, for the ABC data set, Flow Seeding is approximately 20% more accurate than Uniform Seeding, while using 20% less data storage.

Tornado — For this real-world data set, the Flow Seeding approach performs better than the Uniform Seeding approach for every configuration. In Figure 5c, we observe that Flow Seeding is on average 27% more accurate than Uniform Seeding, while being initialized with half the data storage target. Taking data storage costs into account, we can say that Flow Seeding is 27% more accurate on average while using $\frac{1}{4}^{th}$ the amount of data (i.e., one fourth the number of basis flows).

In Figure 5c, the Flow Seeding approach has comparable accuracy to Uniform Seeding, when initialized to use a quarter of the data storage target. For example, Flow Seeding initialized with 1X data storage target, has similar accuracy to Uniform Seeding using 4X data storage. Taking data storage costs into account, Flow Seeding can achieve the same accuracy as Uniform Seeding, while using $\frac{1}{8}^{th}$ the amount of data (i.e., one eigth the number of basis flows).

Thus, when considering data storage and accuracy results for all data sets, Flow Seeding was able to achieve better accuracy for same or reduced data storage, and same accuracy for reduced data storage, than the Uniform Seeding approach.

6.3. Computation Time

We measure the computation time of each phase, i.e., in situ and post hoc, to evaluate performance. For the Flow Seeding approach, the simulation overhead (in situ phase) consists of calculation of basis flows and performing steps involved for particle addition and removal. For the Uniform Seeding approach, the simulation overhead only consists of calculating basis flows. For the post hoc phase, for each interpolation step, we measure the time taken to perform a triangulation to identify particle neighborhoods and to perform particle trajectory interpolations. Computation time of any operation (in situ or post hoc) is dependent on the amount of data being operated on. For each data set, we select configurations representative of the trends observed for computation times.

Double Gyre — Table 2 shows the results for the Double Gyre data set. From Section 6.1 we know that Flow Seeding on average used more data storage than Uniform Seeding. When N_C is high (and P is low), Flow Seeding used more data storage than Uniform Seeding. When N_C is low (and P is high), Flow Seeding used less data storage than Uniform Seeding. For all N_C considered, Flow Seeding used more data storage on average. When comparing post hoc interpolation times, Flow Seeding was between 4x and 35x faster than Uniform Seeding for $N_C = 8$. However, given the increased data storage costs for high N_C, Flow Seeding was between 8x and

Configuration		Uniform Seeding			Flow Seeding		
N_C	Data Storage Target	Sim Overhead	Post Hoc	**Total**	Sim Overhead	Post Hoc	**Total**
2048	1X	0.88	20.35	**21.23**	1.83	267.81	**269.64**
	8X	1.54	43.68	**45.22**	3.05	363.44	**366.49**
128	1X	1.66	5.28	**6.94**	2.28	17.17	**19.45**
	8X	7.39	34.37	**41.76**	24.84	22.73	**47.57**
8	1X	7.16	4.52	**11.68**	37.01	1.11	**38.12**
	8X	40.04	73.95	**113.99**	1911.33	2.17	**1913.5**

Table 2: Timing results for the Double Gyre data set and X = 131,072 points. All timing measurements are reported in seconds.

Configuration		Uniform Seeding			Flow Seeding		
N_C	Data Storage Target	Sim Overhead	Post Hoc	**Total**	Sim Overhead	Post Hoc	**Total**
2048	1X	1.87	1141.85	**1143.72**	20.70	927.46	**948.16**
	8X	8.63	1565.21	**1573.84**	195.54	1161.21	**1356.75**
128	1X	15.38	135.29	**150.67**	105.42	70.10	**175.52**
	8X	80.84	548.74	**629.58**	1386.54	134.85	**1521.39**
8	1X	145.76	61.18	**206.94**	256.34	18.06	**274.40**
	8X	891.52	455.05	**1346.57**	2569.54	129.82	**2699.36**

Table 3: Timing results for the ABC data set and X = 2.1M points. All timing measurements are reported in seconds.

Configuration		Uniform Seeding			Flow Seeding		
N_C	Data Storage Target	Sim Overhead	Post Hoc	**Total**	Sim Overhead	Post Hoc	**Total**
128	1X	149.82	1659.43	**1809.25**	8255.41	247.65	**8503.06**
	8X	1044.15	13701.6	**14745.75**	79909.46	1872.28	**81781.74**
32	1X	527.60	1732.32	**2259.92**	5402.78	161.10	**5563.88**
	8X	4476.33	17331.6	**21807.93**	75245.11	1614.49	**76859.6**
8	1X	1702.15	1697.15	**3399.3**	3735.39	162.39	**3897.78**
	8X	12395.2	13542.2	**25937.4**	67951.8	1473.55	**69425.35**

Table 4: Timing results for the Tornado data set and X = 67.2M points. All timing measurements are reported in seconds.

10x slower than Uniform Seeding for $N_C = 2048$. The simulation overhead of Flow Seeding is greater in every instance resulting in the overall computation time also being greater.

ABC — Table 3 shows the results for the ABC data set. From Section 6.1 we know that Flow Seeding has lower data storage costs than Uniform Seeding for this data set. Consequentially, Flow Seeding requires less post hoc interpolation computation time than Uniform Seeding for every configuration. The simulation overhead of Flow Seeding is greater than Uniform Seeding in every instance. Even though the simulation overhead is greater, when the number of post hoc interpolation steps is large (for example, $N_C = 2048$), Flow Seeding has better total computation time than Uniform Seeding.

Tornado — Table 4 shows the results for the Tornado data set. From Section 6.1 we know that Flow Seeding stores approximately half the data compared to Uniform Seeding. This results in post hoc interpolation computation time being between 6x and 11x faster for Flow Seeding. However, the in situ phase operates on a large number on particles resulting in Flow Seeding having significantly higher simulation overhead.

Overall, the total computation time required by the Flow Seeding approach is greater due to the greater simulation overhead.

7. Conclusion

In this paper, we introduced an alternate approach for using Lagrangian analysis in situ. Our approach is an end-to-end in situ Lagrangian technique, consisting of two phases, which together minimize error propagation and accumulation. For the first phase, we presented an in situ seed placement strategy to generate variable duration basis flows. Our seed placement strategy introduced new particles to prevent poor coverage of the flow field and removed particles to handle dense clustering of particles. For the second phase, we presented a post hoc interpolation technique to make optimal usage of the basis flows generated in situ. Our findings showed that, for all data sets considered, Flow Seeding achieved better accuracy than Uniform Seeding, while using the same or less number of basis flows. Further, Flow Seeding demonstrated the potential for reduced data storage costs depending on the nature of the flow.

While our study did not focus on reducing data storage, and instead focused on minimizing error propagation by putting an upper bound on interpolation error, we believe there is potential for future research to look at various options for controlling the number of basis flows present at any given time. Additionally, our study did not focus on optimizing the performance of the Flow Seeding

approach. We expect parallel algorithms for operations involving the Delaunay triangulation to significantly improve the overall performance of the Flow Seeding approach. We aim to pursue these directions of research in the future.

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.

[ACJC15] AGRANOVSKY A., CAMP D., JOY I., CHILDS H.: *Subsampling-based compression and flow visualization*. Tech. rep., Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA (United States), 2015.

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

[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.

[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.

[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.

[CCK07] CHEN Y., COHEN J., KROLIK J.: Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1448–1455.

[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.

[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.

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HANS H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1464–1471.

[GLT*09] GARTH C., LI G.-S., TRICOCHE X., HANSEN C. D., HAGEN H.: Visualization of coherent structures in transient 2d flows. In *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 1–13.

[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.

[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.

[HE06] HELGELAND A., ELBOTH T.: High-quality and interactive animations of 3d time-varying vector fields. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1535–1546.

[HKTH16] HADJIGHASEM A., KARRASCH D., TERAMOTO H., HALLER G.: Spectral-clustering approach to lagrangian vortex detection. *Physical Review E 93*, 6 (2016), 063107.

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

[HY00] HALLER G., YUAN G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena 147*, 3 (2000), 352–370.

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

[JL00] JOBARD B., LEFER W.: Unsteady flow visualization by animating evenly-spaced streamlines. In *Computer Graphics Forum* (2000), vol. 19, Wiley Online Library, pp. 31–39.

[KHH12] KASTEN J., HOTZ I., HEGE H.-C.: On the Elusive Concept of Lagrangian Coherent Structures. In *Topological Methods in Data Analysis and Visualization II. Theory, Algorithms, and Applications. (TopoInVis'11)* (2012), pp. 207–220.

[KRWT12] KUHN A., RÖSSL C., WEINKAUF T., THEISEL H.: A benchmark for evaluating ftle computations. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE* (2012), IEEE, pp. 121–128.

[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.

[LHS08] LI L., HSIEH H.-H., SHEN H.-W.: Illustrative streamline placement and visualization. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific* (2008), IEEE, pp. 79–86.

[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.

[LS07] LI L., SHEN H.-W.: Image-based streamline generation and rendering. *IEEE Transactions on Visualization & Computer Graphics*, 3 (2007), 630–640.

[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.

[MCHM10] MARCHESIN S., CHEN C.-K., HO C., MA K.-L.: View-dependent streamlines for 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1578–1586.

[MJL*13] MCLOUGHLIN T., JONES M. W., LARAMEE R. S., MALKI R., MASTERS I., HANSEN C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013), 1342–1353.

[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.

[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.

[MLZ09] MCLOUGHLIN T., LARAMEE R. S., ZHANG E.: Easy integral surfaces: a fast, quad-based stream and path surface algorithm. In *Proceedings of the 2009 Computer Graphics International Conference* (2009), ACM, pp. 73–82.

[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.

[Nes89] NESE J. M.: Quantifying local predictability in phase space. *Physica D: Nonlinear Phenomena 35*, 1-2 (1989), 237–250.

[OHBKH09] OBERMAIER H., HERING-BERTRAM M., KUHNERT J., HAGEN H.: Volume deformations in grid-less flow simulations. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 879–886.

[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.

[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.

[SGSM08] SALZBRUNN T., GARTH C., SCHEUERMANN G., MEYER J.: Pathline predicates and unsteady flow structures. *The Visual Computer 24*, 12 (2008), 1039–1051.

[SP07] SADLO F., PEIKERT R.: Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1456–1463.

[STH*09] SHI K., THEISEL H., HAUSER H., WEINKAUF T., MATKOVIC K., HEGE H.-C., SEIDEL H.-P.: Path line attributes-an information visualization approach to analyzing the dynamic behavior of 3d time-dependent flow fields. *Topology-Based Methods in Visualization II* (2009), 75–88.

[SW10] SADLO F., WEISKOPF D.: Time-dependent 2-d vector field topology: An approach inspired by lagrangian coherent structures. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 88–100.

[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).

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 453–460.

[TWHS05] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *Visualization and Computer Graphics, IEEE Transactions on 11*, 4 (July 2005), 383–394. doi:10.1109/TVCG.2005.68.

[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.

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

[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.

[WS05] WIEBEL A., SCHEUERMANN G.: Eyelet particle tracing-steady visualization of unsteady flow. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 607–614.

[YKP05] YE X., KAO D., PANG A.: Strategy for seeding 3d streamlines. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 471–478.

[YWSC12] YU H., WANG C., SHENE C.-K., CHEN J. H.: Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1353–1367.

[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.