ESTABLISHING THE VIABILITY AND EFFICACY OF IN SITU REDUCTION VIA LAGRANGIAN REPRESENTATIONS FOR TIME-DEPENDENT VECTOR FIELDS

by

SUDHANSHU SANE

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

June 2020

DISSERTATION APPROVAL PAGE

Student: Sudhanshu Sane

Title: Establishing the Viability and Efficacy of In Situ Reduction via Lagrangian Representations for Time-Dependent Vector Fields

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Hank Childs	Chair
Boyana Norris	Core Member
Brittany Erickson	Core Member
Leif Karlstrom	Institutional Representative

and

Kate Mondloch	Interim Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded June 2020

DISSERTATION ABSTRACT

Sudhanshu Sane

Doctor of Philosophy

Department of Computer and Information Science

June 2020

Title: Establishing the Viability and Efficacy of In Situ Reduction via Lagrangian Representations for Time-Dependent Vector Fields

Exploratory visualization and analysis of time-dependent vector fields or flow fields generated by scientific simulations is increasingly challenging on modern supercomputers. One possible solution is the use of a Lagrangian-based in situ reduction and post hoc exploration approach. Although this approach offers improved accuracy-storage propositions, prior work has failed to evaluate the viability and efficacy of this method at scale. Additionally, there is a lack of understanding surrounding best practices that advance the effectiveness of the Lagrangian-based approach. This dissertation contributes empirical studies measuring absolute error, calculating the practical in situ encumbrance, and understanding tradeoffs involving accuracy, storage, and performance. Further, this dissertation proposes algorithms that 1) improve accuracy-storage propositions via improved in situ seed placement and post hoc interpolation, and 2) achieve scalability via a communication-free model. Overall, the research presented in this dissertation establishes the viability and efficacy of using Lagrangian representations extracted in situ for post hoc exploratory visualization of large time-dependent vector fields.

This dissertation includes previously published co-authored material.

CURRICULUM VITAE

NAME OF AUTHOR: Sudhanshu Sane

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
University of Pune, Pune, India

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2020,
University of Oregon
Master of Science, Computer and Information Science, 2016,
University of Oregon
Bachelor of Engineering, Information Technology, 2013,
University of Pune

AREAS OF SPECIAL INTEREST:

Flow Visualization
High Performance Computing
Scientific Visualization

PROFESSIONAL EXPERIENCE:

Graduate Research Fellow, University of Oregon, 2016 – Present
Graduate Teaching Fellow, University of Oregon, 2014 – 2016
Graduate Intern, Los Alamos National Laboratory, Summer 2017
Graduate Intern, Lawrence Livermore National Laboratory, Summer 2016

GRANTS, AWARDS AND HONORS:

Area Exam, Passed With Distinction, University of Oregon, 2019
Sushil Jajodia Scholarship, University of Oregon, 2014 – 2019

General University Scholarship, University of Oregon, 2017 – 2019
Kenneth S. Ghent Scholarship, University of Oregon, 2015

PUBLICATIONS:

Sudhanshu Sane and Hank Childs. "In Situ Vector Field Data Reduction Via Lagrangian Representations on Supercomputers." (In submission)

Sudhanshu Sane, Abhishek Yenpure, Roxana Bujack, Matthew Larsen, Kenneth Moreland, Christoph Garth, and Hank Childs. "Scalable In Situ Lagrangian Flow Map Extraction: Demonstrating the Viability of a Communication-Free Model." (In preparation)

Sudhanshu Sane and Hank Childs. "In Situ Lagrangian Analysis for Exploration of Time-Dependent Vector Fields." In Situ Visualization for Computational Science (ISVFCS). (Accepted for publication)

Sudhanshu Sane, Roxana Bujack, Christoph Garth, and Hank Childs. "Survey of Seed Placement and Streamline Selection Techniques," Computer Graphics Forum (CGF). (Accepted for publication)

Sudhanshu Sane, Hank Childs, and Roxana Bujack. "An Interpolation Scheme for VDVP Lagrangian Basis Flows," In Proceedings of EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV), 2019.

Sudhanshu Sane, Roxana Bujack, and Hank Childs. "Revisiting the Evaluation of In Situ Lagrangian Analysis," In Proceedings of EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV), 2018.

Allen D. Malony, Matthew Larsen, Kevin Huck, Chad Wood, **Sudhanshu Sane**, and Hank Childs. "When Parallel Performance Measurement and Analysis Meets In Situ Analytics and Visualization," In Proceedings of the International Conference on Parallel Computing (ParCo), 2019.

Hank Childs, et al. "In Situ Terminology Project," In Proceedings of The International Journal of High Performance Computing Applications (IJHPCA). (Accepted for publication)

Shaomeng Li, **Sudhanshu Sane**, Leigh Orf, Pablo Mininni, John Clyne, and Hank Childs. "Spatiotemporal Wavelet Compression for Visualization of Scientific Simulation Data," In Proceedings of IEEE International Conference on Cluster Computing (CLUSTER), 2017.

Chad Wood, **Sudhanshu Sane**, Daniel Ellsworth, Alfredo Gimenez, Kevin Huck, Todd Gamblin, and Allen D. Malony. "A Scalable Observation System for Introduction and In Situ Analytics," In Proceedings of Workshop on Extreme-Scale Programming Tools (ESPT), 2016.

Pengju Ren, Xiaowei Ren, **Sudhanshu Sane**, Michel A. Kinsy, and Nanning Zheng. "A Deadlock-Free and Connectivity-Guaranteed Methodology for Achieving Fault-Tolerance in On-Chip Networks," In Proceedings of IEEE Transactions on Computers (TC), 2015.

ACKNOWLEDGMENTS

First and foremost I would like to acknowledge my advisor Hank Childs and thank him for being a fantastic and tremendously supportive advisor. I would also like to thank my dissertation advisory committee for their feedback and support.

I owe a great deal to my mentors and colleagues. Matthew Larsen, Roxana Bujack, Christoph Garth, and all my CDUX team members — your help and guidance has been invaluable.

Lastly, I would like to thank my family and friends. They have always given me something to smile about. Of course, last and certainly not least, my cats (Ellie, Biggie, Gucci, and Shorty) and Ashley, for being the best company during this journey.

To my mother, for believing in me every step of the way and teaching me to have faith in hard work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Some of the text in this chapter comes from a manuscript in submission, which was a collaboration between Hank Childs and myself. The text used is relevant for providing context and background information.

1.1 Motivation

The topic of this dissertation is at the intersection of scientific visualization and high-performance computing.

Scientific visualization is a field within computer science that focuses on visualizing and analyzing data produced by scientific simulations. Scientific visualization plays a key role in the scientific discovery pipeline, for understanding, verifying, and exploring scientific data.

High-performance computing is a field within computer science that endeavors to provide more compute capability than a typical desktop computer, and does so via "supercomputers." These machines are particularly useful for computational simulations, which model and study various phenomena of interest. Scientific simulation codes typically utilize high resolution discretized meshes in the domain, where each point in the mesh has multiple field values associated with it (for example, temperature, pressure, density, velocity, etc.) in order to accurately model the phenomena. Each simulation cycle represents the state of the simulation at a specific simulation time and simulation codes could require thousands of cycles to complete.

Combining these two topics (scientific visualization and high-performance computing) causes unique challenges. Executing large scientific simulations often requires hundreds of processors in a distributed memory environment. Further,

these simulations are capable of generating very large amounts of data each cycle. As a consequence, techniques that facilitate and perform data analysis and visualization for scientists must be capable of operating on large amounts of scientific data and running efficiently on high-performance computing resources.

Flow visualization is an important sub-area of scientific visualization for understanding vector field data. Most flow visualization techniques involve placing massless particles at seed positions and displacing them according to a vector field, whether for animating particles, plotting the entire trajectory at once (streamlines/pathlines), or using particle trajectories as building blocks for other techniques (e.g., finite-time Lyapunov exponents). Most often, the movements of particles are calculated by solving an ordinary differential equation, typically with a Runge-Kutta algorithm [1]. Algorithms like Runge-Kutta require evaluating the velocity field at multiple locations; the popular fourth order algorithm (RK4) requires evaluation at four locations. The nature of the evaluations are different based on whether or not the flow field changes as time evolves (unsteady-state flow) or is unchanged as time evolves (steady-state flow). When dealing with steady-state flow, the velocity field evaluations can be performed accurately, with error typically only arising from interpolation within a cell of a mesh. However, when dealing with unsteady-state flow, accurate velocity field evaluations can be harder to achieve.

The primary problem with accurate velocity field evaluation for unsteady-state flow is that computational simulations are unable to store full spatio-temporal data. These simulations often advance in "cycles." During a cycle, the simulation advances from its current time, T, to a new time, $T + \epsilon$. Simulations run for many cycles, from thousands to hundreds of thousands. Saving simulation state to disk often is very costly, both in the time to interact with the I/O system and in

storage costs (bytes). In response, simulations almost uniformly practice "temporal subsampling," meaning that they save data from only a subset of their cycles to disk. With respect to unsteady-state flow, this means that velocity field evaluation must do temporal interpolation; further, as fewer and fewer cycles are stored, these interpolations are increasingly inaccurate, introducing increasing error into flow visualizations.

In this thesis, we consider a class of flow visualization problems with three specific properties. These properties are:

1. The flow visualization is exploratory, i.e., the desired particle trajectories will be specified by a domain scientist during an interactive visualization session after the simulation completes. As a result, the desired particle trajectories are not known while the simulation is running.
2. The flow visualization needs to consider unsteady-state flow (the vector field data from one cycle will not suffice).
3. The simulation saves its data to disk at a low rate, i.e., sparse temporal subsampling.

We refer to this as an **EUS** problem: **E**xploratory analysis + **U**nsteady-state flow + **S**parse temporal subsampling. We note that removing any of these three properties simplifies the problem substantially: **US** can calculate particle trajectories *in situ* while the simulation is running, **ES** does not require inferring velocity field values between cycles, and **EU** can infer velocities between time slices (reasonably) accurately.

EUS problems have occurred more often on supercomputers in recent years. Over the last decade, the ability to generate data on supercomputers has gone up by ∼100X, but I/O capabilities have gone up by ∼10X. As a result,

simulations must the reduce proportion of the data they store, often creating **S**parse temporal sampling. Additionally, while supercomputers are a major motivator for considering **EUS** problems, these problems also are important in non-supercomputing environments.

In situ processing [2, 3] is an important approach for addressing the "I/O bottleneck." Most typically, *in situ* processing utilizes *a priori* knowledge of which visualizations and analyses to complete. However, this *in situ* style is not congruent with the **E**xploratory component of **EUS** problems. Fortunately, an alternate worklow avoids the need for *a priori* knowledge, by using a combination of *in situ* and *post hoc* processing [4]. In the *in situ* phase, data is transformed and reduced so that it is small enough to be saved to disk. In the *post hoc* phase, this data is used to perform exploratory visualization. In the context of flow visualization, this means that *in situ* processing should transform and reduce time-varying vector field data such that *post hoc* exploration can use the result to infer arbitrary particle trajectories — ideally with high accuracy and requiring little data storage, among other properties.

An important consideration with this *in situ* + *post hoc* workflow is how to transform and reduce data. For our work, we transform/reduce spatio-temporal vector data using a Lagrangian approach. This choice enables data from all cycles of a simulation to be represented, which is fundamentally different than the traditional choice of saving time slices. We refer to this Lagrangian-based workflow as **L-ISR-PHE**: **L**agrangian-based **I**n **S**itu **R**eduction with **P**ost **H**oc **E**xploration.

Agranovsky et al. [5] used **L-ISR-PHE** and demonstrated significantly improved storage-accuracy propositions compared to using the traditional Eulerian

paradigm. The method proposed by Agranovsky et al. was straightforward and simple to implement. The simplicity of the method enabled good domain coverage but did not consider the benefits more advanced schemes could offer. The most significant shortcoming of the work was missing evidence that particle trajectory integration could be performed within in situ constraints in a distributed memory environment at scale. As a result, there is a lack of understanding of the technical performance characteristics and the encumbrance placed on a simulation code in a practical setting. Further, the study itself incorporated a "theoretical in situ environment," i.e., an in situ environment was mimicked by loading data from disk. Finally, the study presented accuracy relative to the traditional approach and failed to provide a clear understanding of absolute error, per particle outcomes, and qualitative differences.

1.2 Research Goals and Approaches

This dissertation presents research in response to a central question — "*Is in situ reduction and post hoc exploration via Lagrangian representations a viable alternative to traditional approaches at scale? What practices advance its effectiveness?*"

In particular, it is our goal to conduct studies to address three key issues, i.e., research gaps (**RG**), that prior work did not consider for the **L-ISR-PHE** paradigm:

1. **RG1:** What is the in situ encumbrance? Is it acceptable?
2. **RG2:** Need for greater understanding of accuracy-performance tradeoffs.
3. **RG3:** Is it worthwhile to pursue more advanced schemes?

We identify the following research subquestions (**RQ**) to investigate the corresponding research gaps. To address each of these

- **RQ1.1:** What is the practical in situ encumbrance of the technique?
- **RQ1.2:** Are there ways to reduce in situ encumbrance?
- **RQ2.1:** Are results maintained over varying data sets? Especially non-analytical data sets?
- **RQ2.2:** Is this improvement necessary when considering absolute error?
- **RQ2.3:** What is the spectrum of outcomes compared to the traditional approach?
- **RQ3.1:** Can we benefit by innovating past the simple scheme?
- **RQ3.2:** What new challenges are created by these advancements and can they be addressed?

The following four sections briefly summarize the four main studies conducted for this dissertation. Collectively, they address the **RQs**.

1.2.1 Understanding the In Situ Lagrangian Analysis Parameter Space and Evaluation.

Currently, the performance of **L-ISR-PHE** is informed by a single study by Agranovsky et al. [5]. This study presented the accuracy of Lagrangian analysis relative to the accuracy of Eulerian methods for time-dependent vector field visualization. Although insightful and guiding, the study did not consider (1) absolute error introduced by Lagrangian analysis during flow field reconstruction, (2) causes of error and the performance of in situ Lagrangian analysis across a spectrum of configurations given a fixed data

storage budget, and (3) appropriate metrics to evaluate time-dependent vector field reconstruction.

We conduct a study to improve the understanding for each of the points mentioned above to answer **RQ2.1** and **RQ2.2**. Evaluating the absolute error of in situ Lagrangian analysis and understanding the effect of configuration parameters (number of particles to be used and frequency of saving particle locations) aids the decision-making process concerning future research efforts and practical usage. Further, identifying the appropriate metrics for evaluation improve the validity and usefulness of the approach for multiple flow visualization exploration tasks.

1.2.2 Improve Reconstruction Accuracy-Data Storage Propositions.

The straightforward nature of the existing **L-ISR-PHE** approach has potential shortcomings with respect to resource utilization. Thus, it is valuable to investigate if improved accuracy-storage propositions can be achieved by more advanced schemes. Further, it is important to determine at what cost these advanced schemes can be used.

We conduct a study to improve resource utilization and improve accuracy-storage propositions to address **RQ3.1**, **RQ3.2**, and **RQ2.1**. Additionally, we conduct a comprehensive survey of existing advanced strategies for an analagous problem and inform **RQ3.1**.

1.2.3 Improve Scalability of In Situ Extraction.

A key requirement for in situ Lagrangian analysis to be viable is for the analysis routine to fit within in situ constraints. In situ Lagrangian extraction is performed in a tightly coupled manner with the simulation code and it is critical that the analysis routine scales with the simulation. Poor scalability would result in a large and unacceptable encumbrance placed on the simulation code. Prior work

failed to address how these technique would operate in a large distributed memory environment. For **RQ1.1**, **RQ1.2**, and **RQ2.1**, we conduct a study to evaluate an accuracy-performance tradeoff by considering a communication-free model that offers improved scalability.

1.2.4 Evaluation of Viability of L-ISR-PHE for Large Time-Dependent Vector Fields.

To address **RQ1.1**, **RQ1.2**, **RQ2.1**, **RQ2.2**, and **RQ2.3**, we consider large time-dependent vector fields generated by Exascale Computing Project (ECP) applications. We consider hydrodynamics, cosmology, and seisomology applications. To evaluate viability, we measure in situ execution costs (communication, particle advection, seed placement), I/O costs (write time, bytes), and reconstruction accuracy in comparison with the traditional Eulerian approach. Further, we test these methods for high resolution simulations runs in a distributed memory environment on the supercomputer Summit at ORNL.

1.3 Dissertation Outline

This disseration is organized as follows:

Part I — Foundations

- Chapter II provides relevant, foundational background information on the topics of high-performance computing and in situ processing.
- Chapter III provides a full description of Lagrangian analysis in the context of an in situ data reduction operator. Further, it describes how this approach differs from the traditional approach.

Part II — Findings

– Chapter IV presents our study that broadens the evaluation of spatiotemporal tradeoffs, causes of error, and appropriate evaluation metrics for **L-ISR-PHE**.

– Chapter V introduces a new approach called "variable duration, variable placement" (VDVP). This chapter explores the use of long trajectories to improve accuracy, and presents a flexible interpolation scheme to reduce post hoc interpolation error.

– Chapter VI introduces a novel communication-free model for in situ Lagrangian basis flow extraction, in order to improve scalability.

– Chapter VII presents our evaluation of **L-ISR-PHE** on simulation codes in practical settings.

Part III — Conclusion and Future Work

– Chapter VIII provides a synthesis of the findings and recommendations for future work.

– Chapter IX is a survey that informs future efforts. Specifically, it surveys selection of representative sets of integral curves, since these techniques may inspire new approaches for in situ Lagrangian representation extraction.

1.4 Co-Authored Material

Much of the research presented in this dissertation is from previously published co-authored material. Below is a listing connecting the chapters with the publications and the authors that contributed to it. Further detail concerning the division of labor is provided at the beginning of each chapter. That being said,

for each of these works, I was not only the first author of the paper, but also the primary contributor for implementing systems, conducting studies, and writing manuscripts.

- Chapter III: Contains text from a publication [6] that was a collaboration between Hank Childs and myself.
- Chapter IV: Contains text from a publication [7] that was a collaboration between Roxana Bujack, Hank Childs, and myself.
- Chapter V: Contains text from a publication [8] that was a collaboration between Roxana Bujack, Hank Childs, and myself.
- Chapter VI: Contains text from a manuscript in preparation [9] and was a collaboration between Abhishek Yenpure, Matthew Larsen, Kenneth Moreland, Christoph Garth, Roxana Bujack, Hank Childs, and myself.
- Chapter VII: Contains a text from a manuscript in submission and is a collaboration between Hank Childs and myself.
- Chapter IX: Contains text from a publication [10] that was a collaboration between Christoph Garth, Roxana Bujack, Hank Childs, and myself.

Part 1

Foundations

CHAPTER II

BACKGROUND

Some of the text in this chapter comes from a work in preparation [9]. The contributors for this text were Hank Childs and myself. The text used is relevant for providing context and background information.

This short chapter introduces the high-performance computing context and current trends that impact the data analysis and visualization community and thus, the scientific community. Next, it discusses the role in situ processing is playing in responding and adapting to the current trends and constraints. Finally, this chapter highlights the additional benefits that can be gained from operating in situ.

2.1 High-Performance Computing and Scientific Visualization

High-performance computing (HPC) systems are large computing platforms that scientists leverage to gain insight in fields including cosmology, combustion, seisomology, medicine, energy systems, and national security. Scientists use computational models of various phenomena to study and improve our understanding of these complex systems. Data analysis and visualization plays a critical role for researchers in the scientific discovery process.

The HPC ecosystem is constantly evolving, and up until the past decade, the nearly standard approach of performing scientific analysis and visualization for computational simulations executing on high-performance computing resources was "post hoc" processing. In the post hoc setting, computational simulations first execute on supercomputers and save data to permanent storage. The simulation data is then loaded at a later time for analysis and visualization by scientists. However, the paradigm of only using post hoc processing is under threat because the amount of data generated by computational simulations can be very large.

Machine	FLOPS	I/O	Year of Debut
ORNL: Jaguar	1.75 PetaFLOPs	240 GB/s	2009
ORNL: Titan	27 PetaFLOPs	240 GB/s -> 1 TB/s	2012
ORNL: Summit	187 PetaFLOPs	1 TB/s -> 2.5 TB/s	2018

Machine	FLOPS	I/O	Year of Debut
TACC: Ranger	0.58 PetaFLOPs	500 GB/s	2008
TACC: Stampede	18 PetaFLOPs	1 TB/s	2017

Table 1. Compute and I/O trends on Petascale supercomputers at ORNL and TACC.

Over the last decade, the compute capabilities of supercomputers have increased by a whole order of magnitude relative to the I/O capabilities of these machines. Table 1 shows the compute and I/O trends on Petascale computers at ORNL and TACC. As a result of these trends, computational simulations on supercomputers are now able to generate data much faster than they can store it. Although temporal subsampling, e.g., saving every N^{th} time step for subsequent *post hoc* visualization, has always been practiced, the inherent I/O bottleneck on modern supercomputers is forcing this subsampling to become severe. Such sparse temporal settings can hinder accurate reconstruction and exploration by domain scientists during post hoc processing.

2.2 In Situ Processing

In situ processing [3], i.e., coupling analysis routines with the simulation code and processing data as it is generated, addresses this issue by limiting I/O. A significant benefit of operating in situ is access to all the spatio-temporal data generated by the simulation. This creates opportunities that were not available

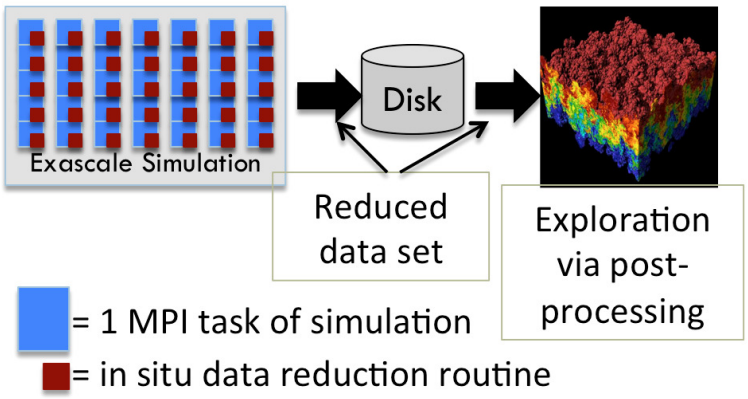


Figure 1. Data analysis and visualization workflow incorporating both in situ and post hoc processing. Image courtesy of Childs [4].

when storing only a fraction of time slices to disk and is particularly consequential for analysis of transient or time-varying phenomena in scientific simulations. In situ processing has gained significant momentum in the last half-decade, including myriad research efforts, devoted workshops (ISAV, WOIV), and a recent Dagstuhl seminar [11, 12].

A key issue for in situ processing, particularly in **scientific exploration use cases**, is whether there is a priori knowledge on precisely which analysis and visualizations are desired. If a priori knowledge exists, then in situ analysis and visualization is a matter of integrating the desired routines into a simulation code. If not, then in situ visualization is more complicated, since it is unclear what activities to do. A common solution to this problem is to transform the data to a reduced form, small enough that it can be stored to disk for post hoc exploration. We refer to this paradigm as "in situ reduction - post hoc exploration," or ISR-PHE. Typically, the ISR-PHE paradigm involves navigating a tradeoff between data integrity and data reduction, and ensuring that the corresponding routines

operate within in situ constraints. In this dissertation project, we consider time-varying flow visualization in the context of ISR-PHE.

Figure 1 illustrates the ISR-PHE workflow from generation of scientific data to its consumption. In the ISR-PHE workflow, in situ routines are integrated and executed on either the same compute resources ("a tightly coupled setting") or on distinct resources ("a loosely coupled setting") as the simulation code. In this dissertation, we will limit our scope to the scenario with shared compute resources. When an in situ routine is operating in tandem with a simulation, control is handed to the in situ routine at some specified frequency. The in situ routine will typically be passed the relevant simulation data for it to operate on. The routine completes its computation — it might generate some output data products in the form of visualizations, derived data, statistics, etc — and returns control to the simulation code. This dissertation focuses on generating a reduced data representation of the simulation's time-varying vector field.

Once an in situ routine produces a reduced data set and stores it to disk, scientists can employ various data analysis and visualization tools to explore their data at their convenience. The key opportunity that in situ processing plays is the generation of a reduced data set that reduces strain on the storage system and can be more amenable for interactive post hoc exploration, while maintaining high data integrity.

Chapter III discusses in detail a Lagrangian-based in situ data reduction operator for time-varying vector field data.

CHAPTER III

IN SITU LAGRANGIAN ANALYSIS FOR EXPLORATION OF TIME-DEPENDENT FLOW FIELDS

Most of the text in this chapter comes from a publication [6], which was a collaboration between Hank Childs and myself. I was responsible for preparing and writing the majority of the publication. Hank Childs provided extensive feedback during the work and was involved in editing the manuscript.

In this chapter, we discuss the approach for enabling post hoc exploration of time-dependent vector fields via in situ data reduction from the Lagrangian perspective. This chapter informs a variety of considerations around this topic, including fundamental concepts, how it works, constraints, differences from the traditional method, and more.

3.1 Introduction

Flow fields produced by computational fluid dynamics (CFD) simulations often require scientific visualization to understand, verify, and explore phenomena of interest. These flow fields are typically represented as time-dependent vector fields defined over discretized meshes, and these meshes are often very high resolution in order to achieve accurate modeling. When doing flow visualization, individual algorithms can choose to operate on the vector field data as "steady state" (i.e., ignoring that the vector field evolves over time, typically in order to reduce execution time by considering less data) or as "unsteady state" (i.e., recognizing the time-dependent nature of the field, typically at a cost of higher execution time). That said, for unsteady state flow visualization to be accurate, it requires complete spatiotemporal data. This data rarely exists for post hoc analysis in practice, since simulation codes need to perform temporal subsampling

to fit their results on disk. The result is that unsteady state flow visualizations are inaccurate, and they become increasingly inaccurate as the subsampling gets greater and greater. Worse, trends in high-performance computing are leading temporal subsampling to become increasingly aggressive, leading unsteady state flow visualization to become increasingly inaccurate.

With respect to in situ, important considerations include (1) whether there is a priori knowledge of what to visualize and (2) whether the visualization should be using steady state or unsteady state flow. When there is a priori knowledge, then in situ processing is straightforward: the in situ routines can be executed as the simulation advances, producing the desired results. When the desired visualizations consider steady state flow, then the only data required comes from a single time slice (or a handful of time slices if they want to do steady-state flow at select times during the simulation). In this case, the simulation data for that time slice can be saved to disk, either at full resolution (if possible based on I/O constraints) or via a reduced form. That said, when there is no a priori knowledge of what to visualize (i.e., exploratory analysis) and the desired visualizations must consider unsteady state flow, then traditional approaches for flow visualization break down. *This use case is the purpose for considering the* ***L-ISR-PHE*** *paradigm.*

This thesis focuses on one way to approach time-dependent vector field data reduction: in situ processing to calculate a reduced Lagrangian representation of a time-dependent vector field. Seminal research in the field demonstrated the approach provides improved accuracy and data storage propositions compared to traditional methods [5]. Figure 2 shows the relation of the in situ Lagrangian

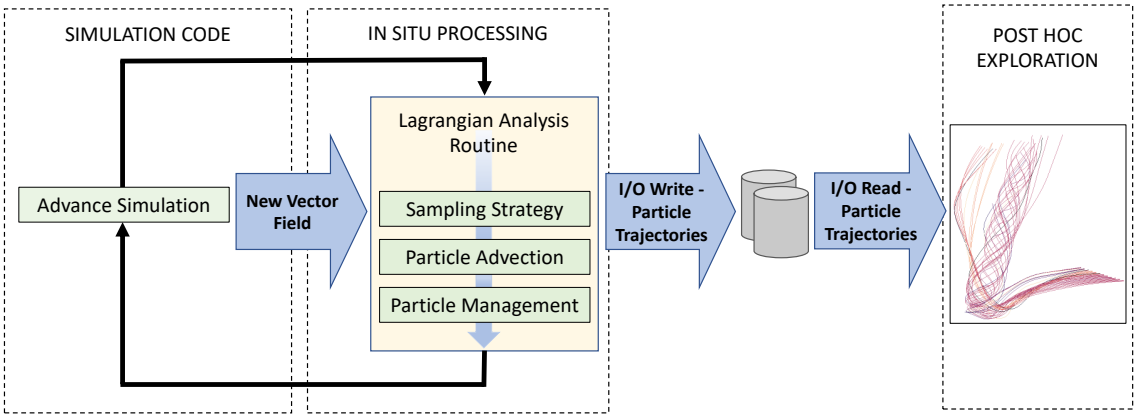


Figure 2. Diagram showing the operations of an in situ Lagrangian routine, as well as its relation to the simulation code and post hoc visualization workflow.

routine to the simulation code and its role in the visualization workflow as a data reduction technique.

3.2 Background and Motivation

This background section contains two parts. Section 3.2.1 discusses how flow fields are specified and specifically the two frames of reference for representing time-dependent vector data: Eulerian and Lagrangian. Section 3.2.2 discusses the limitations of the traditional, Eulerian-based paradigm for visualizing and analyzing time-dependent vector data.

3.2.1 Frames of Reference in Fluid Dynamics.

In fluid dynamics, the Eulerian and Lagrangian frames of reference are two ways of looking at fluid motion. When considering the Eulerian frame of reference, the observer is at a fixed position. In the Lagrangian frame of reference, however, the observer is attached to a fluid parcel and moves through space and time. In computational fluid dynamics, simulations can be designed to employ a fixed mesh (Eulerian), feature simulation grid points that follow the simulation velocity field (Lagrangian), or use hybrid Eulerian-Lagrangian specifications.

When a flow field is stored in an Eulerian representation, it is typically done by means of its velocity field. A velocity field v is a time-dependent vector field that maps each point $x \in \mathbb{R}^d$ in space to the velocity of the flow field for a given time $t \in \mathbb{R}$

$$v : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d,\ x, t \mapsto v(x, t) \tag{3.1}$$

In a practical setting, the vector field is defined over a fixed, discrete mesh and represents the state of the flow field at a specific instant of time or time slice, i.e., at a specific simulation time and cycle.

When a flow field is stored in a Lagrangian representation, it is done by means of its flow map $F_{t_0}^t$. The flow map consists of starting positions x_0 at times t_0 of massless particles and the corresponding particle trajectories advected by the vector field. The mathematical definition of the flow map is the mapping

$$F_{t_0}^t(x_0) : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d,\ t \times t_0 \times x_0 \mapsto F_{t_0}^t(x_0) = x(t) \tag{3.2}$$

of initial values x_0 to the solutions of the ordinary differential equation

$$\frac{d}{dt}x(t) = v(x(t), t) \tag{3.3}$$

In a practical setting, the flow field is stored as sets of particle trajectories calculated in the time interval $[t_0, t] \subset \mathbb{R}$. The stored information, encoded in the form of known particle trajectories, represents the behavior of the flow field over an interval of time.

Finally, the two frames of reference are theoretically equivalent [13], i.e., particle trajectories can be calculated from the vector field through integration and the vectors from the particle trajectories through differentiation. That said, their application in practical contexts varies.

3.2.2 Traditional Paradigm for Visualization and Analysis of Time-Dependent Flow Fields.

Particle advection — calculating the trajectory a massless particle follows in a flow field — is a foundational operation in integration-based flow visualization techniques (line integral convolution, finite-time Lyapunov exponents, pathlines, etc.). Flow visualization tasks are traditionally performed utilizing the Eulerian frame of reference. Given an Eulerian representation of the vector field as input, calculating particle trajectories requires an Eulerian-based particle advection scheme. In general, for a given particle, Eulerian-based particle advection methods require interpolating the velocity of the flow field at the location of the particle. A particle advection operation assumes the availability of the complete spatiotemporal resolution of the flow field as generated by the simulation. However, when considering a time-dependent flow field, only a subset of this data is practically available in a post hoc, Eulerian settings. Thus, the velocity field must be interpolated for time steps in between stored time slices. Interpolation of temporally sparse data, however, can introduce significant error due to numerical approximation. On the one hand, if the vector field does not change significantly, the error will be relatively low. On the other hand, if the time between known steps is large or the vector field changes significantly, the error can be high and features can be missed.

From an implementation perspective, vector field data mapped onto a fixed mesh yields two significant benefits: fast cell location and east interpolation. This makes the computational cost of the particle advection operation during post hoc flow visualization relatively inexpensive.

As discussed in Chapter II, modern supercomputing trends indicate that the computational capacity will continue to outpace I/O capabilities. This means

that I/O operations — writing large files to disk representing simulation output and subsequently reading those files from disk for post hoc exploratory flow visualization — will remain bottlenecks in their respective workflows.

Overall, these challenges in accuracy and performance jeopardize our ability to perform time-dependent flow visualization of large vector fields under the traditional paradigm and motivate new approaches.

3.3 Lagrangian-Based Flow Analysis

In situ processing provides significant opportunities to perform more accurate time-dependent flow visualization, since it can access more spatiotemporal data. In cases where there is a priori knowledge of what to visualize, the traditional, Eulerian paradigm can be used. That said, when there is no a priori knowledge, i.e., for exploratory use cases, the Eulerian paradigm is not as useful — although in situ processing increases the available temporal information, an Eulerian approach cannot exploit this additional data since the particle trajectories desired for a flow visualization are not known. The Lagrangian paradigm, however, can make use of this additional temporal information. Specifically, research has shown that reducing the vector field data by transforming it into its Lagrangian representation and saving this new form to disk for later exploration offers significant benefits.

The following subsections consider the two distinct phases of computation involved in **L-ISR-PHE**, as well as the differences between the Lagrangian and Eulerian paradigm from the perspective of the data analysis and visualization workflow.

3.3.1 Phases of Computation.

There are two phases of computation involved in Lagrangian-based flow analysis: in situ extraction and post hoc

exploration (described in detail in Sections 3.4 and 3.5, respectively). The in situ phase involves calculating a Lagrangian representation of the flow field, i.e., tracing basis trajectories (pathlines that can be used subsequently to infer additional pathlines). Access to the complete spatial and temporal resolution of the flow field allows the in situ extraction routine to accurately calculate the trajectories that form the stored flow map. The goal of the in situ Lagrangian extraction routines is to select trajectories that should be calculated and stored while remaining within in situ constraints. The in situ constraints on Lagrangian extraction are discussed in more detail in Section 3.4.1.

With respect to a Lagrangian representation, the following characteristics are desired:

- It should be possible to compute within in situ constraints.
- It should maximize information per byte stored to disk.
- It should support accurate and interactive post hoc exploration.

The major research challenge with **L-ISR-PHE** is to achieve all three of these characteristics. Further, the basic framework of the Lagrangian paradigm inherently has no constraints on how particle trajectories should be seeded, terminated, selected, represented, or stored. In particular, selection of particle trajectories requires smart sampling along both spatial and temporal axes. These important areas of flexibility are discussed in Sections 3.4.2 and 3.4.3.

The second phase, i.e., the post hoc exploration phase, involves performing flow visualizations using the stored Lagrangian basis trajectories. In this phase, there are no constraints on the types of flow visualizations performed; the flow visualizations specify seed locations, and new pathlines can be interpolated using

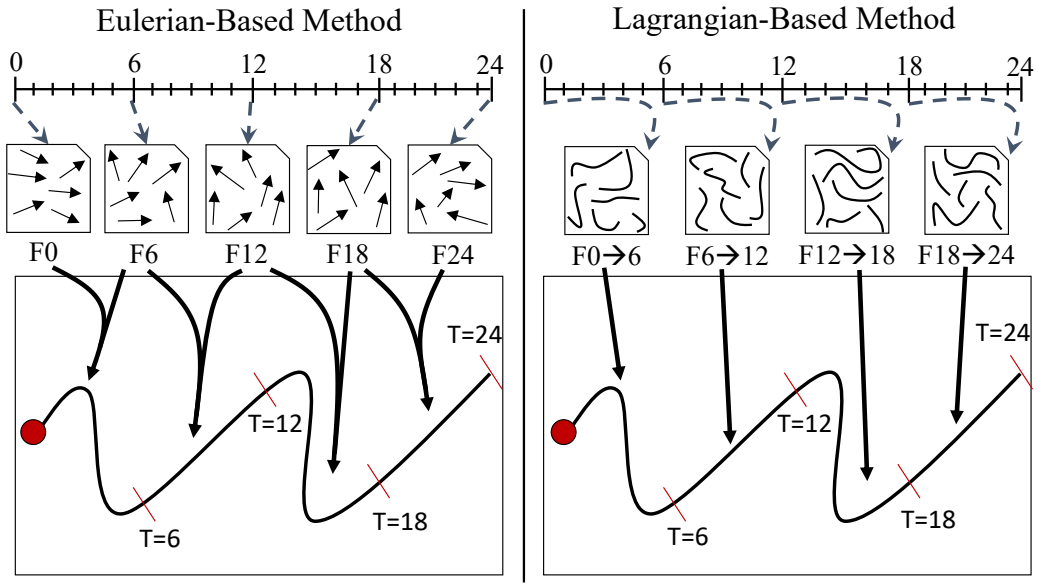


Figure 3. Notional example showing the differences between an Eulerian-based and a Lagrangian-based method. There are differences in the type of data stored, the information represented by the data, and what data is interpolated post hoc to calculate a new particle trajectory. In particular, the Eulerian-based method stores time slices, while the Lagrangian-based method stores time intervals. Image recreated and based on original figure by Agranovsky et al. [5].

the in situ extracted data. This interpolation step is non-trivial and also the subject of research (discussed more in Section 3.5). That said, one important approach is to incorporate principles from scattered point interpolation methods.

3.3.2 Differences between Eulerian and Lagrangian-Based Flow

Analysis. Unlike the Eulerian paradigm, the Lagrangian paradigm operates in two distinct phases of computation. The first phase, i.e., in situ processing, introduces an encumbrance on the simulation code. The impact of the overhead is an important consideration to evaluate the practicality of the method.

To better highlight the differences between the methods, consider a notional example. Figure 3 presents an example where a simulation code runs for 24 cycles total and outputs data every 6 cycles. The Eulerian method stores time slices,

denoted by FX, where X denotes the cycle. The Lagrangian method calculates particle trajectories in situ and the stored data captures the behavior of the flow field over an interval of time. These files are denoted by F$X \rightarrow Y$, where X and Y represent the cycles that begin and end the interval of calculation. Eulerian-based advection occurs by solving ordinary differential equations and performing spatial and temporal interpolation. In this case, two time slices (files) are used for particle advection. For Lagrangian-based advection, a single file is used to advance a particle across an interval of time. Although the new, interpolated particle trajectory is identical in both cases of this notional example, it will likely be different in practice. Further, in practice, the Lagrangian trajectory is typically superior, since the stored Lagrangian representation can encode more information per byte of storage, enabling high accuracy. Section 3.6 contains visualizations highlighting the differences in the interpolated trajectories in practical settings.

Table 2 summarizes some differences between the two methods. Although the Lagrangian-based method has an increased encumbrance from running in situ, preliminary work has showed the tradeoffs in storage requirements, accuracy, and post hoc performance are superior [5]. With respect to reconstruction accuracy, the Eulerian approach incurs increasing error as temporal sparsity increases, while the Lagrangian method has increasing error when the number of basis trajectories reduces. Under sparse temporal settings, the Lagrangian-based method has been demonstrated to be up to 10X more accurate than the Eulerian-based method when considering the same storage, and can provide comparable accuracy for data reductions of up to 64X. Further, depending on the data reduction, the reduced burden on I/O write and read operations can result in significant performance improvements.

Table 2. Differences between the Lagrangian and Eulerian paradigm from a simulation's perspective. This table is based on one appearning in work by Agranovsky et al.[5].

	Eulerian	Lagrangian
Saved files contain	Vector fields	Particle trajectories
Saved files represent	Time slices	Time intervals
Reducing I/O, storage	Less time slices	Less particles
Increasing accuracy	More time slices	More particles
Simulation overhead	I/O	I/O + Lagrangian analysis routine computation and memory

3.4 In Situ Extraction

This section discusses the in situ extraction phase of **L-ISR-PHE** in more detail. Specifically, the section begins with a discussion of the in situ costs and constraints. Next, it covers strategies for spatial and temporal sampling of a time-dependent flow field. The section concludes with a discussion of options for storage of a Lagrangian representation and the impact it can have on the overall workflow.

3.4.1 In Situ Costs and Constraints. **L-ISR-PHE** introduces a new cost that does not occur with the traditional paradigm: the cost of calculating basis trajectories. Fortunately, there are several parameters or "knobs" that exist within the Lagrangian framework that can impact (and potentially reduce) these costs. The five main components that contribute to in situ costs are:

- The particle advection workload, i.e., the number of particle trajectories being calculated.
- The frequency and amount of I/O required by the in situ routine.

- Communication costs to exchange information in a distributed memory setting.
- Computation costs of a sampling strategy.
- Costs of storing particle trajectory information in memory.

These components can directly affect in situ costs, for both memory usage and execution time. In addition, in situ analysis routines are often allocated a limited resource budget and then required to operate within this budget. The limited available memory places restrictions on (1) the number of particle trajectories that can be calculated, (2) the number of locations along each particle trajectory that can be stored, and lastly, (3) the number of time slices that can retained in memory in situ. To remain within in situ constraints, an appropriate particle advection workload and in-memory particle trajectory representation must be selected. In cases where memory usage increases over an interval of computation, the frequency at which data is moved from memory to disk is pertinent.

Computing a Lagrangian representation involves computing integral curves in a distributed memory environment. Although several research works have considered preprocessing of the vector field and redistribution of the data, these options are less feasible within an in situ context. When operating in situ, the vector field decomposition is determined by the simulation. Depending on the distribution of the data and the underlying vector field, the number of particles crossing node boundaries to continue advecting could result in an increase of overall execution time. With respect to a sampling strategy (spatial and temporal), the execution time required will vary depending on the sampling algorithm and

corresponding implementation. These processes should, ideally, utilize the available hardware acceleration on the compute nodes of modern supercomputers to produce low-cost, fast techniques that minimize the overall encumbrance on the simulation code. Overall, in order to not exceed usage of allocated resources, these costs and constraints must be accounted for when designing an in situ Lagrangian extraction routine.

3.4.2 Spatial Sampling: Seed Placement. Spatially sampling the vector field, i.e., selecting locations to seed basis particles, plays a critical role in determining the quality of the data extracted. In general, a spatial sampling strategy is responsible for directing the following three operations:

- An initial placement of seed particles.
- When a new seed particle should be introduced.
- When an existing seed particle should be terminated.

Seed placement or spatial sampling strategies can be guided by the desire to achieve any of the following objectives in varying orders of priority:

- Maximize information content per byte stored to disk.

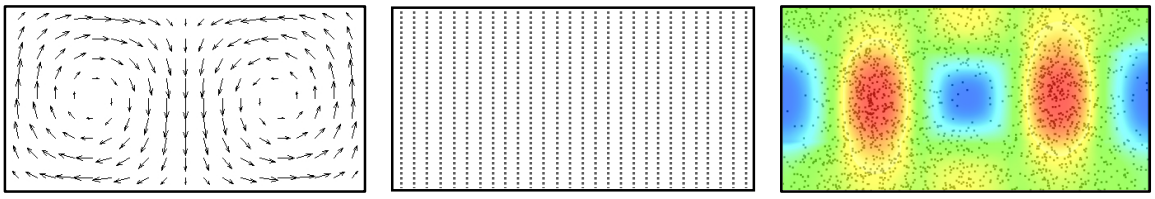


Figure 4. An illustration of two possibilities for seed placement given the Double Gyre vector field. Figure (a) shows a vector glyph visualization. Figure (b) presents a simple uniform seed placement and Figure (c) colormaps an entropy field derived from the vector field and shows seed placement with a density proportional to the value of entropy.

- Coverage of the domain, i.e., every region of the domain receives coverage.
- Focus on capturing regions of interest or specific features accurately.
- Minimizing in situ encumbrance, i.e., fast execution times and/or low memory usage.

To work as an effective in situ data reduction operator, however, these objectives are sometimes in tension and require navigating a tradeoff. Although maximizing the information per byte stored to disk might seem of highest priority, it is easy to imagine a scenario where this characteristic is compromised for better domain coverage or algorithmic simplicity.

Spatial sampling strategies must consider the distribution of seed particles in both space and time. Given the nature of particle trajectories in unsteady state flow, particles can cluster in regions while leaving voids in other regions. To address this problem, seed particles need to either be reset periodically or added/removed as required to maintain coverage. In the study by Agranovsky et al. [5], seeds are placed at uniform locations in the domain and the particle trajectories are terminated at regular intervals. Although this approach can provide good domain coverage using short trajectories, a uniform distribution of particles is not always the best allocation of resources. In Chapter V, we present our own novel algorithm where seed particles follow the flow field and form long trajectories (since post hoc interpolation of long trajectories has been shown to be more accurate [13, 14]), storing locations uniformly along the trajectory. Further, the approach used a Delaunay triangulation over the seeds to (1) identify locations to add seeds and fill voids and (2) remove seeds in regions of seed clustering. Although this approach

provided up to 2X more accuracy using 50% less basis trajectories, the in situ cost of the sampling strategy was increased.

There is much research to be done in this space. For example, a strategy might benefit from seed placement guided by flow field features derived from the vector field (e.g., entropy, curvature, divergence, etc.). Figure 4 illustrates a uniform and entropy-guided seed placement for the Double Gyre vector field (a commonly used analytic data set). Multiple research studies in the area of seed placement [15, 16, 17] demonstrate the efficacy of field-guided approaches for streamline selection, and these findings very well may translate to **L-ISR-PHE**. Further, informative scalar fields can be quickly derived from vector fields when using parallel resources. To summarize, field-guided spatial sampling strategies have significant potential to improve information per byte stored in a Lagrangian representation while remaining computationally efficient.

3.4.3 Temporal Sampling: Curve Approximation. Temporal sampling refers to how much of a single basis trajectory should be stored. A basis trajectory contains the entire route a particle traveled. In practical terms, this route is comprised of the positions resulting from each advection step. The temporal sampling question, then, is which of these positions along the trajectory should be saved? Saving all of the positions along a trajectory will best capture the underlying flow field, but incurs a large storage cost. Saving fewer positions reduces this storage cost, at a tradeoff of reduced accuracy.

For each basis trajectory being calculated in situ, there are multiple options for sampling and storing the trajectory. A straightforward strategy is to save only the start and end points along the pathline computed in situ. Although this strategy can provide data storage optimizations and be sufficient for

approximating the pathline, in the event of a long interval of calculation or complex flow field behavior this strategy could be an oversimplification. Agranovsky et al. [5] calculated short basis trajectories and stored only end locations. Temporal sampling is more relevant when long basis trajectories are being computed in situ. The algorithm in Chapter V calculated basis trajectories of variable length and uniformly sampled the trajectory. Alternate strategies might consider various curve simplification techniques, like selecting points along a pathline that minimize its reconstruction error, or using attributes like curvature, winding angle, or linear and angular entropy to guide a temporal sampling strategy. Figure 5 uses notional examples to illustrate a comparison between uniform and attribute-guided curve sampling.

Complex temporal sampling strategies are challenging due to the limited in situ memory available and the higher memory requirements of these strategies, i.e., the requirement of storing multiple points along a pathline before selecting a subset. That said, collective smart temporal sampling across all basis trajectories could enable a high fidelity reconstruction of the time-dependent flow field for further reducing storage costs.

3.4.4 Storage Format. Decisions for how to carry out spatial and temporal sampling affect the storage layout for basis trajectories. In turn, this can impact storage and memory usage, I/O times (for in situ writing and post hoc reading), and post hoc execution time (for reconstruction and interpolation).

In general, there are two options for storage: structured and unstructured. With a structured data set output, the seed locations of basis trajectories lie along a regular grid. The implicit nature of the grid eliminates the need for storing seed locations explicitly, and promotes a natural organization of data. Trajectory

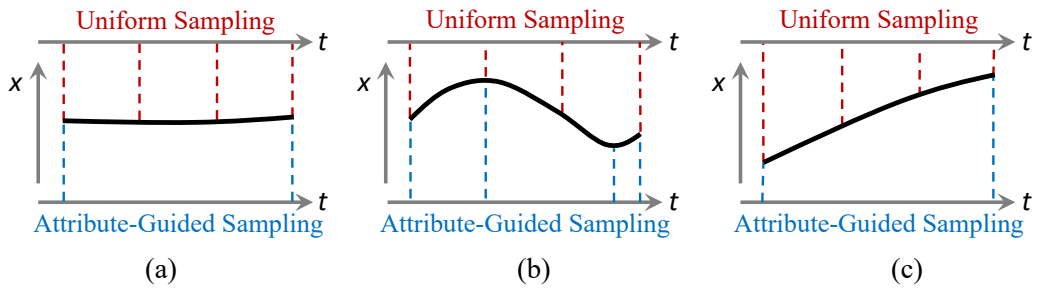


Figure 5. This figure contains three notional basis flows, to illustrate two different curve sampling strategies: uniform and attribute-guided. Each basis flow is colored in black, and the data saved is indicated with dotted red lines for uniform sampling, and dotted blue lines for attribute-guided sampling. Figure (a) considers a curve whose x value remains relatively the same across time. In this case, attribute-guided sampling can reduce storage costs and still provide the same information. Figure (b) shows a curve whose x value first increases, then decreases before increasing again. In this case, attribute-guided sampling can place samples where they will best inform the nature of the trajectory. Figure (c) shows a curve whose x value steadily increases across time. Its benefits are similar to Figure (a) – reducing storage costs.

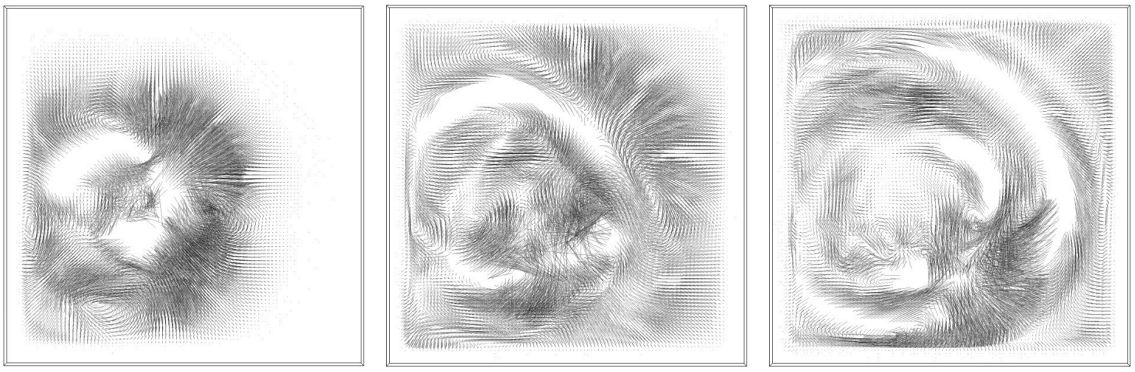
data can be stored in "grid" form, with the values at each grid point representing information about the seed that originated at that location. Agranovsky et al. [5] adopted this approach, and stored only a single value at each grid location: the ending position of the seed particle. Of course, additional data can be stored at each grid point. For example, a field can be used to indicate whether a particle remained within the bounds of the domain during the interval of calculation. Although the use of a structured storage format enables a fast post hoc exploration workflow, it is limited to the case of spatial sampling along a regular grid. Figure 6 shows direct visualizations of a structured data output, where particle trajectories (represented as line segments) are calculated using the displacement field of a seismology simulation over three intervals of computation.

When considering the flexibility the Lagrangian framework provides, the unstructured storage format is a more natural fit when spatial and temporal

sampling are irregular (i.e., adapted to maximize the information per byte). Particle trajectories can be stored as current point locations, lines (two points), or polylines (more than two points), each with additional attributes (for example, an identifier to link data points of the same basis trajectory across files) associated with each object. This approach, however, results in a more expensive post hoc exploration process since cell location and interpolation require more elaborate search structures when considering unstructured data. The algorithm in Chapter V adopted an unstructured storage format to store long particle trajectories, calculated across multiple file write cycles, as line segments with identifiers. Although the adopted approach did not increase in situ costs, post hoc reconstruction required the use of a search structure (Delaunay triangulation) to locate relevant basis trajectories.

3.5 Post Hoc Exploration

After the in situ generation of basis trajectories, post hoc exploration of the time-dependent flow field can be performed with nearly any flow visualization technique. As flow visualization techniques depend on analyzing particle trajectories, the only difference is in how the trajectories are obtained. Where the traditional Eulerian approach calculates the trajectories via particle advection steps, the Lagrangian approach calculates the trajectories by interpolating between nearby basis trajectories. This interpolation can be thought of as "following" basis trajectories, i.e., using them as a guide to infer where particles in between the trajectories would travel. This section discusses how to interpolate a new particle trajectory, as well as the corresponding search structures that can be used to locate nearby basis trajectories.



(a) F3000→4000 (b) F4000→5000 (c) F5000→6000

Figure 6. This figure is an example of directly visualizing the data extracted from an in situ Lagrangian analysis routine. Each image is a visualization of a Lagrangian representation of the time-dependent vector field produced by a seismic modeling simulation studying seismic wave propagation [18]. For this example, seeds are initially placed along a regular grid and only the end points of the particles are saved at the end of an interval, i.e., a structured grid with an "endpoint" field is stored. In each visualization, the trajectories (represented as line segments from the grid point to the "endpoint") capture the displacement caused by the underlying vector field. For example, Figure (a) shows trajectories calculated between cycle 3000 and 4000 and represents the earlier stages of the simulation. The progressive propagation of the seismic waves can be perceived when considering all three images.

To calculate a new particle trajectory starting at a specific location x_i, a "neighborhood" of basis trajectories to follow needs to be identified. Typically, depending on the distances of the neighborhood basis trajectories from x_i and the selected interpolation method, different weights are assigned to each basis trajectory and then used to calculate the next location of the new particle trajectory. Interpolation methods such as barycentric coodinate interpolation, Moving Least Squares interpolation, Shepard's method, etc., can be used to calculate new particle trajectories [19, 20]. A notional example for interpolation is illustrated in Figure 7. The determination of the neighborhood is largely dependent on how the basis trajectories are stored (structured or unstructured). When the Lagrangian representation is stored using a structured data set, the neighborhood

can be identified as the basis trajectories that are initialized at the grid points of the cell containing the location of the new particle trajectory to be interpolated. When using an unstructured data set, the neighborhood can be identified as the set of basis trajectories within a specific search radius or those that form a convex hull that contains the location of the new particle trajectory. Chandler et al. [21] used a modified k-d tree to perform a radius search and Sane et al. [8] used a Delaunay triangulation to identify a containing cell given a Lagrangian representation stored in an unstructured data set. Further, techniques like binning can be used to accelerate the neighborhood identification process.

The result of interpolating basis trajectories is a set of points that form a pathline. These points along the pathline are an interval in time apart. To estimate the position of the particle in between interpolated locations, various curve fitting techniques can be used. A simple and straightforward approach to visualize the

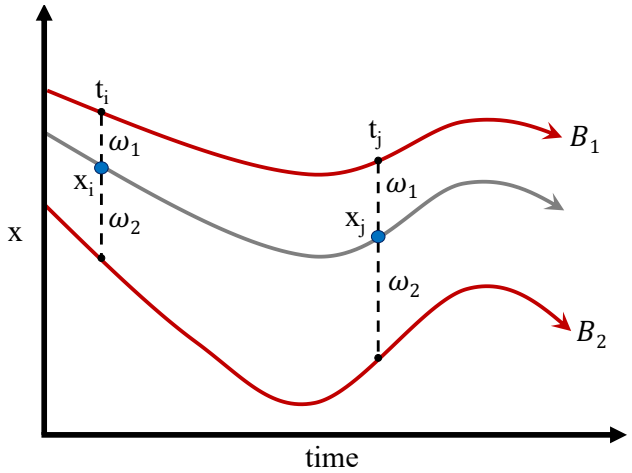


Figure 7. Interpolation of a new particle trajectory (gray) using previously computed basis trajectories (B_1 and B_2 in red). Each position along the new particle trajectory is calculated by following basis trajectories. At time t_i, weights ω_1 and ω_2 are calculated based on the distance of B_1 and B_2 from x_i respectively. The weights are then used to estimate x_j at time t_j. This process can continue to trace the complete trajectory.

interpolated positions along a pathline is the use of a C^0 polygonal chain. This approach is used to visualize pathlines interpolated from basis trajectories in Section 3.6. As the size of the interval increases, however, the aesthetic quality of the C^0 representation of the pathline deteriorates and can be improved by using parameter curves. Bujack et al. [13] studied and evaluated the use of multiple parameter curves (cubic Hermite spline, Bèzier curve) to represent particle trajectories.

The complexity of a post hoc Lagrangian-based interpolation routine is dependent on the format of the extracted Lagrangian representation of the time-dependent vector field. If basis trajectories are long and span across several simulation cycles storing multiple positions along the way, then pathline interpolation following the same neighborhood of basis trajectories results in more accurate interpolation [13, 8]. Although following short (single interval) basis trajectories is straightforward, changing the neighborhood frequently propagates a local truncation error [13, 14, 7]. Overall, there is a need for complex, yet efficient, and accurate post hoc Lagrangian-based interpolation systems, and this topic requires future research.

3.6 Efficacy of Lagrangian-Based In Situ + Post Hoc Flow Analysis

For time-dependent flow visualization, the Lagrangian paradigm offers significantly improved accuracy and data storage propositions compared to the Eulerian paradigm under sparse temporal settings. This is possible because the Lagrangian representation of time-dependent vector data is capable of encoding more information per byte. The Lagrangian representation captures the behavior of the underlying flow field over an interval of time. This is in contrast to an Eulerian representation that captures the vector data at a single time slice. Further, in

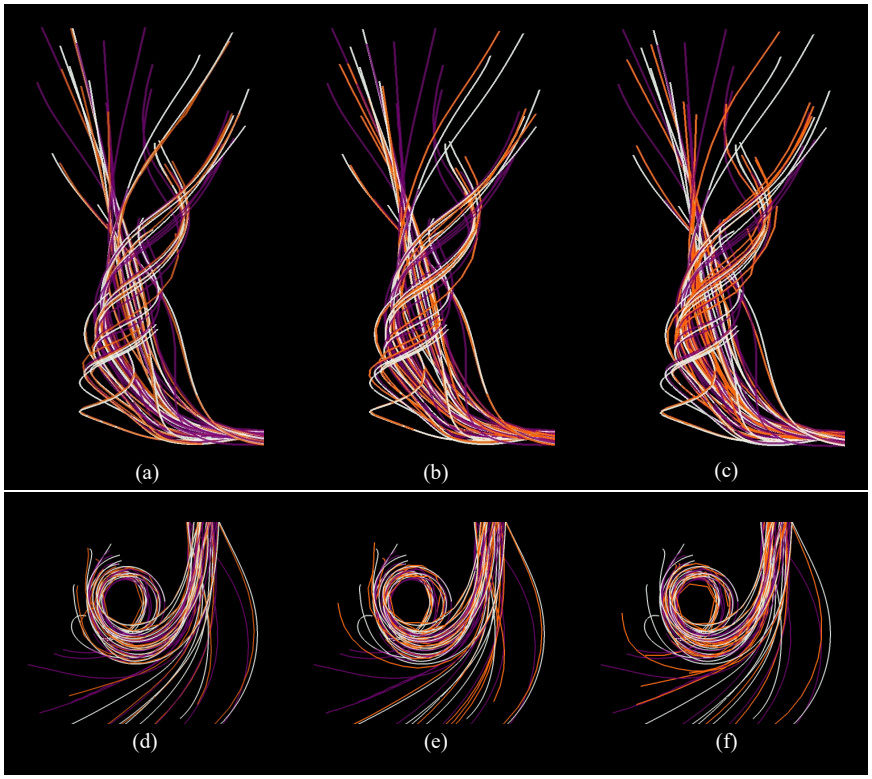


Figure 8. Visualizations of an F-5 tornado vortex to qualitatively compare the accuracy of pathlines traced using two methods under sparse temporal settings: Lagrangian (orange trajectories) and Eulerian (purple trajectories). The ground truth set of trajectories, calculated using every cycle of the simulation, is traced in white. The six visualizations present results for varying configurations of how many basis trajectories are stored in the Lagrangian representation. The configurations are: 1:1 ratio of particles to grid points in (a) and (d), 1:2 in (b) and (e), and 1:4 in (c) and (f). With respect to temporal subsampling, every 8^{th} simulation cycle is stored. For all three configurations, the method described by Agranovsky et al. [5] is used. For the post hoc visualizations, particles are initially seeded in a rake and trace trajectories that enter the tornado vortex region from the bottom-right (a-c) and top-right (d-f) of the figures. In Figures (a) and (d), the white (ground truth) and orange (Lagrangian) pathlines following very similar trajectories. The Eulerian method pathlines (purple) have diverged from the white trajectories in most instances. Figures (b) and (e) show the Lagrangian representation performing better than the Eulerian method using half as much storage. A similar trend can be observed in Figures (c) and (f) where Lagrangian configurations use a quarter of the storage. In these figures Lagrangian accuracy deteriorates as the number of particle trajectories stored reduces from 1:1 to 1:4. That said, in all cases the accuracy of the Lagrangian pathlines remains higher than the corresponding Eulerian pathlines using the full spatial resolution.

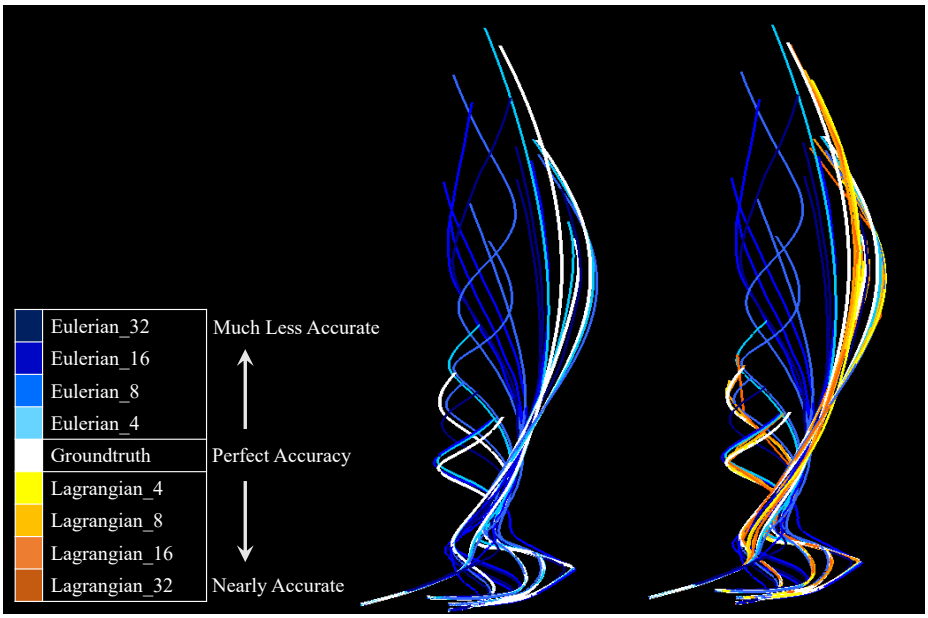


Figure 9. Pathline visualizations to compare the accuracy of pathlines traced using two methods: Eulerian and Lagrangian. Lagrangian_X and Eulerian_X denote configurations that store data every X^{th} cycle, i.e., the size of the interval. Left: five ground truth pathlines and the corresponding Eulerian pathlines. Right: five ground truth pathlines, and the corresponding Eulerian and Lagrangian pathlines.

situ access to the complete spatiotemporal resolution of the simulation vector field enables accurate computation of the Lagrangian representation. This section highlights the efficacy of the Lagrangian-based approach for time-dependent flow visualization by considering both quantitative and qualitative aspects.

The study by Agranovsky et al. [5] demonstrated the ability of the Lagrangian representation to retain substantially better accuracy relative to the Eulerian method, even with significantly less data. For example, the study demonstrated the Lagrangian-based approach achieving comparable accuracy using a 64X data reduction. When using the same amount of storage as the Eulerian approach, the Lagrangian representation enabled an over 10X more accurate reconstruction of the flow field. Further, the study showed how increasing the interval between storing information to disk is far less detrimental to the

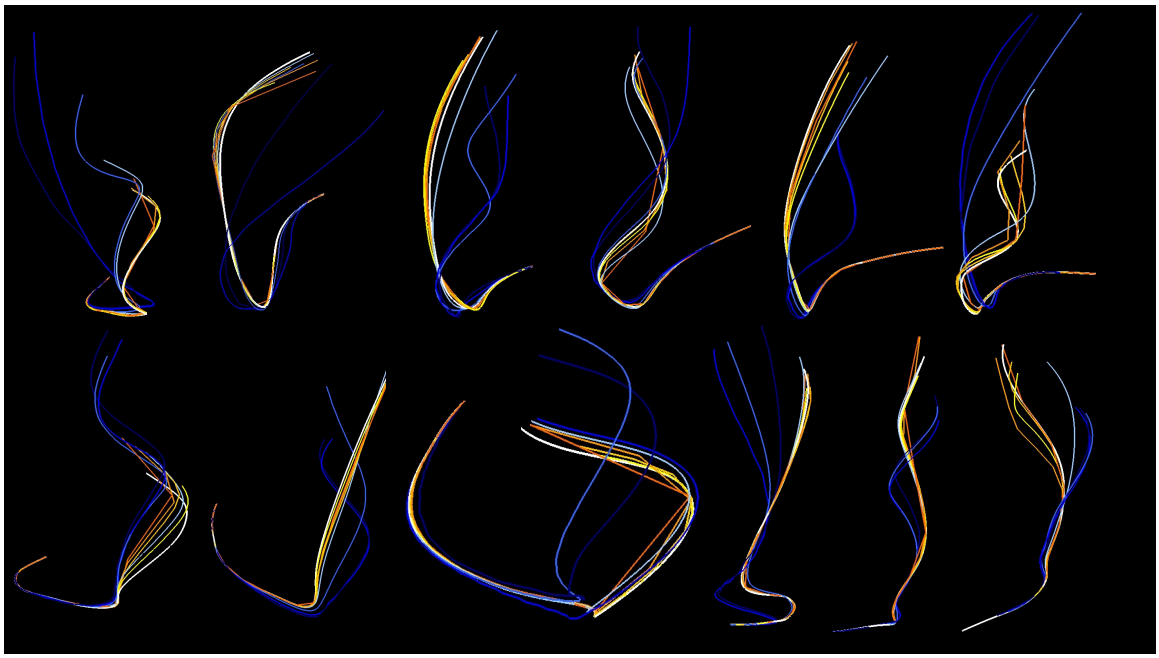


Figure 10. Twelve sets of pathlines traced using multiple Eulerian and Lagrangian configurations. Each set shows a ground truth pathline, and four Eulerian and Lagrangian pathlines each. The color scheme is the same as that described in Figure 9. Overall, Lagrangian pathlines more closely follow their respective ground truth trajectories and the Eulerian pathlines are less accurate in settings of temporal sparsity, i.e, configurations with large intervals between saving data to disk.

Lagrangian-based method than the Eulerian approach. We complement and broaden the Agranovsky study with our own study in Chapter IV. Chapter IV presents a study using multiple evaluation metrics to compare the absolute errors of both the methods and observe trends across a range of spatiotemporal configurations. This study also demonstrated the significantly improved accuracy-storage propositions offered by the Lagrangian method in settings of temporal sparsity.

To compare the Lagrangian and Eulerian methods qualitatively, Figures 8, 9, and 10 visualize pathlines from an F-5 tornado weather simulation. The simulation has a base grid resolution of $490 \times 490 \times 280$. Further, we consider

512 consecutive simulation cycles, corresponding to the time period when a mature tornado vortex exists. Ground truth pathlines that are interpolated using every simulation cycle, and Lagrangian and Eulerian trajectories computed under sparse temporal settings, are visualized. In Figure 8, Lagrangian pathline interpolation using the same number of particles as grid points (Figure 8a) is nearly perfectly accurate to the ground truth. As the number of particles used reduces (Figures 8b and 8c), the accuracy of Lagrangian pathline interpolation decreases. That being said, it remains more accurate than the Eulerian method. In comparison to the Lagrangian pathline interpolation, the pathlines generated by the Eulerian method trace much less accurate trajectories. Figures 9 and 10 demonstrate the effects of increasing temporal sparsity on the Lagrangian and Eulerian methods. As expected, when the size of the interval increases, the accuracy of the Eulerian pathlines decreases. In nearly every case, only the Eulerian pathline computed using a small interval size remains accurate. In contrast, the Lagrangian pathlines remain nearly accurate and closely follow the ground truth trajectories irrespective of the size of the interval.

Overall, the Lagrangian method is capable of significantly improving our ability to perform exploratory time-dependent flow visualization by providing high integrity reconstructions of the flow field and requiring less data to be extracted from a simulation.

3.7 State of the Art of In Situ Lagrangian Analysis

Over the past decade, Lagrangian methods have been increasingly used for flow visualization. Lagrangian coherent structures (LCS), proposed by Haller et al. [22, 23, 24], are a popular technique to visualize attracting and repelling

surfaces and have seen research focused on the acceleration of computation and visualization [25, 26, 27, 28], and the application of LCS [29, 30, 31].

In a post hoc setting, Hlawatsch et al. [32] explored the use of a hierarchical scheme to construct longer pathlines using previously computed Lagrangian basis trajectories. The constructed pathlines would be more accurate due to being constructed using fewer integration steps.

In recent years, there has been an increased interest in the use of in situ processing to compute a Lagrangian representation of the flow field, i.e., **L-ISR-PHE**. Agranovsky et al. [5] first demonstrated the benefits of this approach. Chandler et al. [21] extracted a Lagrangian representation from an SPH [33] simulation and used a modified k-d tree to accelerate post hoc interpolation. Additionally, Chandler et al. [34] conducted studies to identify correlations between Lagrangian post hoc interpolation error and divergence in the flow field. Other theoretical error analysis studies and empirical evaluations have been conducted to study the absolute error of Lagrangian analysis and error propagation during particle trajectory computation [13, 14, 7]. Most recently, Sane et al. [8] explored the use of variable duration basis trajectories and proposed interpolation schemes to accurately compute pathlines by following long basis trajectories to reduce error propagation.

Part 2

Findings

This part of the dissertation presents the findings and the details of our research studies. Chapter IV presents our study that improves our understanding of the spatiotemporal tradeoffs, causes of error, and appropriate evaluation metrics for **L-ISR-PHE**. This initial study served to cultivate our understanding of the research space, and to empirically understand post efficacy characteristics and evaluation methods. Next, Chapter V presents research considering novel approaches to improve accuracy-storage propositions compared to the current state of the art. The study builds upon theoretical understandings regarding causes of error when interpolation Lagrangian representations. We propose a method that reduces error and supports more flexibility with respect to sampling the domain. Chapter VI explores the use of a communication-free model to address scalability during in situ Lagrangian basis flow extraction. This is an important advancement in order to make in situ encumbrance low enough that the **L-ISR-PHE** paradigm can be viable. Chapter VII presents our evaluation of **L-ISR-PHE** for simulation codes on a supercomputer. This empirical study provides the first clear evidence that a Lagrangian representation can be viably extracted at scale while supporting high integrity post hoc efficacy for real-world simulations.

CHAPTER IV

REVISITING IN SITU LAGRANGIAN ANALYSIS

Most of the text in this chapter comes from a publication [7], which was a collaboration between Roxana Bujack, Hank Childs, and myself. I was responsible for system implementation, conducting the experiments, preparing, and writing the manuscript. Roxana Bujack advised and assisted with theoretical background in the manuscript. Hank Childs advised, provided extensive feedback during the work and was involved in editing the manuscript.

4.1 Introduction

The main purpose of this chapter is to further evaluate the **L-ISR-PHE** paradigm. Although the initial study by Agranovsky et al. [5] showed the potential of the technique, their evaluation had two significant issues. First, all of their results were comparative in nature. There was no information provided regarding spatial and temporal tradeoffs. In short, it showed Lagrangian techniques were superior to Eulerian techniques, but did not provide insights into how many basis flows were needed to achieve desired accuracies. Second, their accuracy metric focused on the end location of an interpolated particle trajectory, and did not consider the locations between the seed and the end point. This resulted in a limited overview of the accuracy of particle trajectories as a whole, and in particular for circular flow.

This study addresses both of the issues with their evaluation. The result both supplements the evaluation of Agranovsky et al. and also provides new understanding of the efficacy of their technique. We believe the evaluations in the current study will be the most useful comparators for future Lagrangian research

that endeavors to improve on the work of Agranovsky et al. Specifically, our study focuses on the following points:

- We use an accuracy metric which evaluates the entire particle trajectory.
- Where the previous study considered reduced storage for only the Lagrangian approach, our study considers reduced storage for both approaches.
- We conduct experiments evaluating interpolation steps, which advance a particle forward in time. Specifically, we study the effect of large numbers of interpolation steps, each of which results in further advancement in time. This is important because each interpolation step has an associated error, and so multiple interpolation steps suffer from error propagation and accumulation.

4.2 Theoretical Background

In this section, we will provide a brief recap of the theoretical foundations. We use h_t to denote the resolution in time and h_x for the resolution in space.

Post-hoc advection in the Eulerian setting is typically performed using the fourth order Runge Kutta scheme, which is an iterative numerical integration method that has a total accumulation error of $O(h_t^4)$ [35, 36]. Since we use it on top of discrete data interpolated multi-linearly in space, it actually is of the overall order $O(h_t^4 + h_x^2)$.

Previous work [13] showed that the Lagrangian method as described by Agranovsky et al. [5] is also a numerical integration method with a total accumulation error of $O(h_x^2)$ for each interpolation. Since we approximate the intermediate values between two cycles using linear interpolation in time, it is of overall order $O(h_t^2 + h_x^2)$.

4.3 Study Overview

4.3.1 Study Configuration. We use the same in situ basis flow extraction and post hoc pathline interpolation technique as Agranovsky et al. [5]. Like the Agranovsky study, our in situ environment was theoretical, evaluating analytic data sets on the fly or loading simulation results from disk.

4.3.1.1 Data Sets. We used the following data sets for our experiments:

Double Gyre — This data set is a two-dimensional flow field consisting of two counter-rotating gyres with a time dependent perturbation. This data set is simulated for 1024 cycles at a base grid 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 [37]. This data set is simulated for 400 cycles with a base grid resolution of $128 \times 128 \times 128$. We set the period of the ABC flow to 100 cycles.

Tornado — This data set is from a simulation of the dynamics of an F5 tornado [38]. The base grid resolution is $490 \times 490 \times 280$. A mature tornado vortex exists in the domain during the 512 simulation seconds we considered for our experiments. Our collaborating scientist normally uses a temporal frequency of "every two simulation seconds" for his studies. Thus, we consider 256 time slices, with the time-steps evenly distributed from $t_0 = 8502$s to $t_{256} = 9014$s.

Let N_T denote the total number of time slices or cycles. For the Double Gyre data set, $N_T = 1024$. For the ABC data set, $N_T = 400$. For the Tornado data set, $N_T = 256$.

4.3.1.2 Storage Budget.

We use the term storage budget to denote the allowed amount of data that can be saved to disk for post hoc pathline interpolation. We believe allowing both Lagrangian and Eulerian the same storage budget enables a fair comparison between them. Let N_C denote the number of cycles saved ($N_C \leq N_T$) and let P denote the number of basis flows or vector samples stored per cycle. If B denotes the storage budget for total number of basis flows or vector samples that can be saved, we select combinations of N_C and P such that $N_C \times P = B$. Further, we set the value of B to be equal to the number of points in the base grid resolution of each data set respectively. For the Double Gyre data set, $B = 131,072$ points. For the ABC data set, $B = 2.1$M points. For the Tornado data set, $B = 67.2$M points. For our experiments we consider three storage budgets ($1B, 2B, 4B$) for each data set. For each budget, we select multiple configurations that are combinations of N_C and P.

For example, the Double Gyre $2B$ test used 262,144 points. It further varied N_C with values ranging from 4 to 1024. For $N_C = 4$ we calculate four intervals of basis flows (Lagrangian) or four time slices of vectors (Eulerian), with each set containing $P = 65536$. Similarly, for $N_C = 1024$, there are 1024 sets, with $P = 256$.

4.3.2 Error Evaluation.

For a given seed point, we calculate its corresponding pathline using three different methods.

- **Ground Truth** — We calculate the ground truth trajectories with a fourth-order Runge Kutta scheme [1] using the full spatial and temporal resolution available for each data set. The ground truth is considered to be perfectly accurate and have 0% error.

– **Lagrangian** — We calculate basis flows in situ with each selected configuration of N_C and P. We then use the calculated basis flows post hoc to interpolate new Lagrangian trajectories.

– **Eulerian** — We calculate Eulerian trajectories with each selected configuration of N_C and P, for comparison with the Lagrangian approach. Similar to ground truth calculation, a fourth-order Runge Kutta scheme is used to calculate the Eulerian trajectories.

Together, these three sets of trajectories can be used to evaluate and compare the approaches.

For both the ABC and Double Gyre data sets we randomly seed 1000 points over the entire flow field. For the Tornado data set, we place 144 seeds along rakes at locations used by our collaborating scientist to study the phenomena.

In contrast to the error metric used by Agranovsky et al. [5], we use a standard curve evaluation error metric — the L2-norm. The number of positions of a particle to represent the ground truth is equal to N_T. However, the number of known positions for a Lagrangian trajectory is N_C.

Given a test configuration average L2-norm is calculated as follows:

$$\frac{1}{p}\sum_{i=0}^{p}\frac{1}{n}\sum_{t=0}^{n}||x_{i,t}-g_{i,t}|| \tag{4.1}$$

where p is the total number of particles, $x_{i,t}$ is the location of a Lagrangian or Eulerian interpolated particle i at time t and $g_{i,t}$ is the location of the ground truth particle i at time t. We use two variants of the L2-norm to calculate the error:

– **Full L2-Norm Metric** When calculating the Full L2-norm, n is equal to N_T (total number of cycles).

– **Select L2-Norm Metric** When calculating the Select L2-norm, n is equal to N_C (number of cycles saved).

Agranovsky et al. used $n = 1$, which is similar in spirit to the Select L2-Norm; we add the Full L2-Norm for our evaluation to capture behavior along the interpolated trajectory at locations between the seed and the end point.

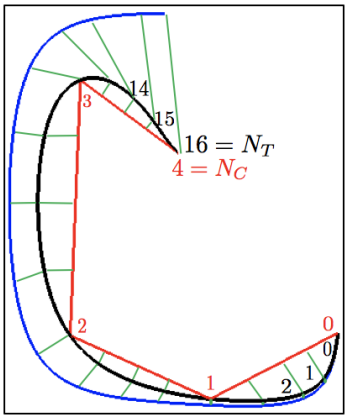


Figure 11. Notional example of trajectories interpolated using Lagrangian-based (red) and Eulerian-based (blue) methods and the corresponding ground truth (black).

Figure 11 illustrates a notional example of the difference between the accuracy metrics for a simplified sample trajectory. While the Lagrangian trajectory is accurate at its known points, the remainder of the trajectory can significantly deviate from the ground truth because it is linearly interpolated from the known points. We expect the Full L2-norm evaluation to show this error for low N_C configurations. In contrast, the Select L2-norm, which evaluates only at the known points along the trajectory, shows how close a particle is to the ground truth at these locations. Together, these error metrics provide a more comprehensive evaluation and understanding of a Lagrangian trajectory accuracy as a whole.

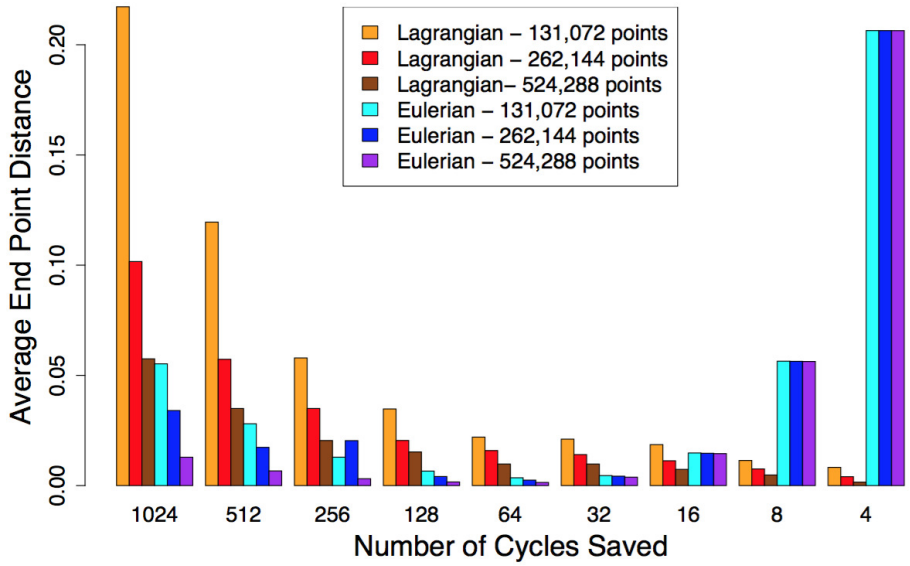


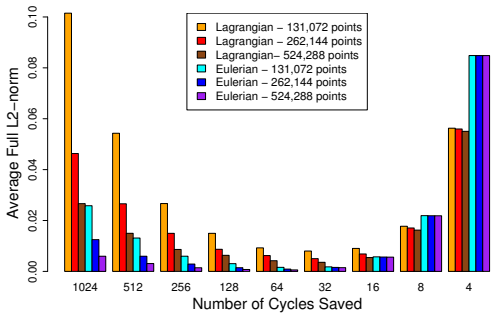
Figure 12. Double Gyre analysis verifying Agranovsky et al.'s results.

4.4 Results

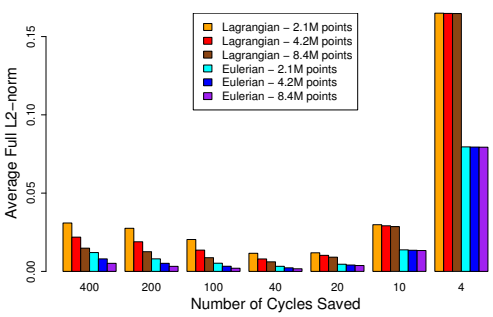
The first step of our evaluation was to verify the Agranovsky et al. results using their error metric. One of our results, for Double Gyre, is plotted in Figure 12. These results match their findings. Further, the results are similar to the Select L2-norm results in Figure 13b.

After verifying Agranovsky et al.'s results, we proceeded with our own study. The trends we observe, for both our spatiotemporal and error propagation analyses, are consistent regardless of data set. Figure 13 plots our results.

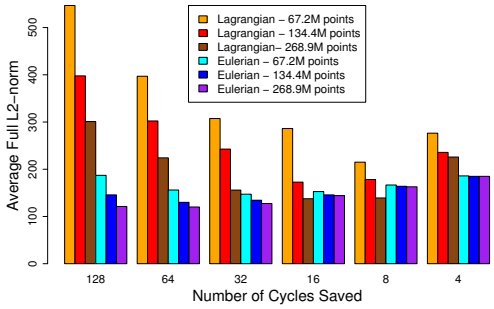
4.4.1 Spatiotemporal Tradeoff. For each of the Full L2-norm evaluations, the optimal values fell in between our largest and smallest N_C configurations (i.e., $N_C = 32$ for Double Gyre, 20 for ABC, and 8 for Tornado). This represents configurations using a sufficiently high P, enabling accurate interpolation, and sufficiently high N_C, such that the trajectory is well represented even with linear interpolation being performed between known points.



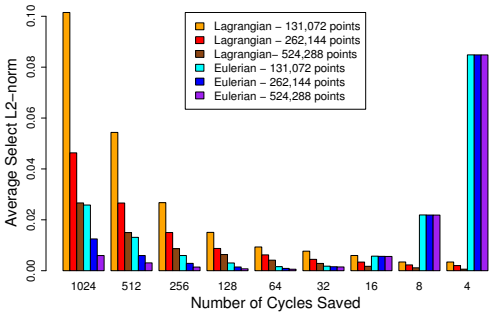
(a) Double Gyre - Full L2-norm



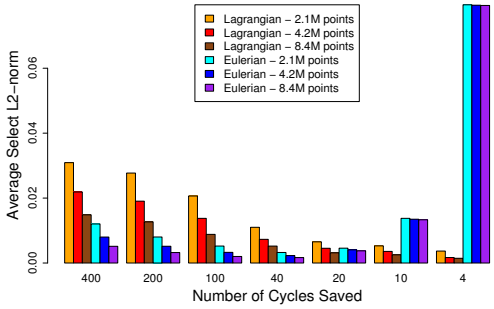
(b) ABC - Full L2-norm



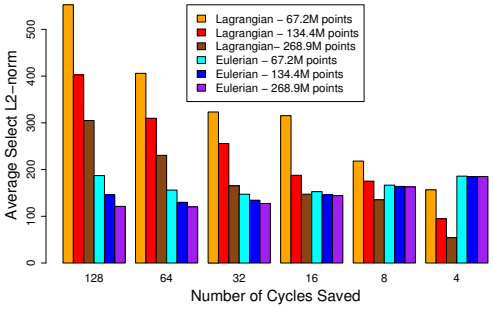
(c) Tornado - Full L2-norm



(d) Double Gyre - Select L2-norm



(e) ABC - Select L2-norm



(f) Tornado - Select L2-norm

Figure 13. Evaluation results for Full L2-norm and Select L2-norm. Legends indicate the total data storage budget information.

As N_C gets smaller, the Lagrangian trajectories have poor accuracy as a whole (see Full L2-norm result) even though the interpolated trajectory follows the ground truth closely at known locations (see corresponding Select L2-norm result). For example, in Figures 13a and 13b, for $N_C = 4$ and 8, we observe high error

for Full L2-norm but low error for Select L2-norm. As demonstrated by Bujack et al. [13], curve fitting can significantly reduce the Full L2-norm error for Lagrangian trajectories. For Eulerian configurations with a low N_C, the approach suffers from low temporal resolution and this is reflected in the high error for both metrics.

Further, we observe increases in storage benefit the Lagrangian approach more than the Eulerian approach. An increase in the number of basis flows reduces the interpolation error per step. Figure 14 shows Double Gyre trajectories for multiple configurations, each using the same total storage.

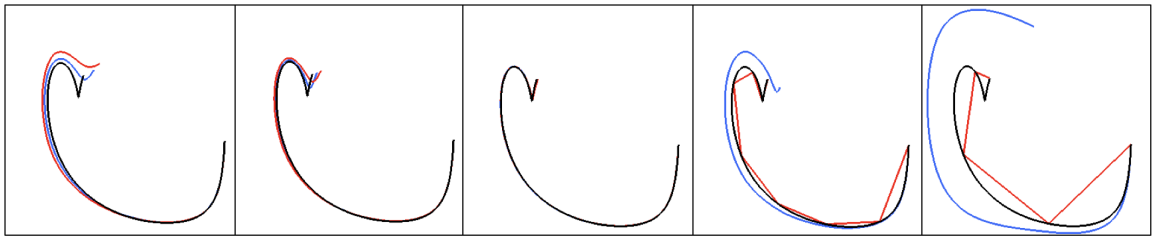


Figure 14. Series of sample trajectories interpolated in the Double Gyre data set using varying number of cycles saved. Color code: Black - Ground Truth, Red - Lagrangian, Blue - Eulerian. From l-r : 1024, 512, 64, 8 and 4 cycles saved.

4.4.2 Error Propagation. High N_C configurations in particular allow us to study the effect of large numbers of interpolations steps, each of which advances a particle forward in time. The performance of the Lagrangian trajectories for high N_C values is poor relative to the Eulerian approach. The first contributing factor is the low value of P (which, for this study, is inversely proportional to N_C to keep total storage constant). We observe large gains in accuracy with an increase in storage budget for these configurations. The second contributing factor is the error propagation which occurs when using the one-step second order integration method [14] for Lagrangian interpolation. The left-most set of trajectories in Figure 14 show the difference in error accumulation when

using the second order integration method for the Lagrangian trajectory and the fourth order integration method used for the Eulerian trajectory.

4.5 Conclusion

Our study provided information regarding spatial and temporal trade-offs when working with a fixed storage budget. Further, by considering multiple storage budgets, our study informed trade-offs between data reduction and accuracy for the Lagrangian approach. With these results, future researchers can make better informed decisions regarding how many basis flows are needed to achieve reasonably high accuracy. Another takeaway from our study is an increased understanding of best practices for the Lagrangian approach with respect to tradeoffs between number of basis flows and frequency of output. Further, we suggest (and use) two variants of the L2-norm which together provide a more comprehensive evaluation of a Lagrangian trajectory.

CHAPTER V

AN INTERPOLATION SCHEME FOR VDVP LAGRANGIAN BASIS FLOWS

Most of the text in this chapter comes from a publication [8], which was a collaboration between Hank Childs, Roxana Bujack, and myself. I was responsible for system implementation, conducting the experiments, and preparing and writing the manuscript. I did the majority of the work as an intern at Los Alamos National Lab, where Roxana Bujack was my mentor for this project. Roxana Bujack and Hank Childs advised, provided extensive feedback during the work, and were involved in editing the manuscript.

5.1 Introduction

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 [13, 14]. 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 also introduce an interpolation scheme for VDVP flows, VDVP-Interpolation, 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 **L-ISR-PHE** for the **EUS** problem 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 interpolation 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.

5.2 Background and Related Work

5.2.1 Seed Placement Techniques for Flow Analysis.

Chapter IX covers several seed placement and streamline selection research

works. 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. Specifically related to strategies for the extraction of basis flows, Agranovsky et al. [5] 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 5.5). The system we adopt is most similar to Mebarki et al. [39] who used Delaunay triangulation to identify cavities in the field and then placed seeds at the centroid of the triangle.

5.2.2 Fixed Duration Fixed Placement. Summarizing discussion from Chapter III, Agranovsky et al. [5] 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.

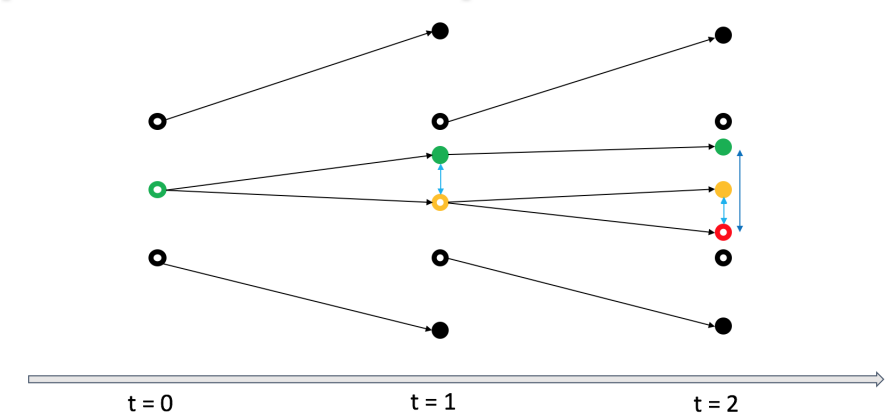
In the second phase, the basis flows from the first phase are used to approximate the behavior of the flow field. To begin, for a given particle, the algorithm identifies a neighborhood of surrounding basis flows to follow. Specifically, the neighborhood is the set of basis flows that form a minimum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

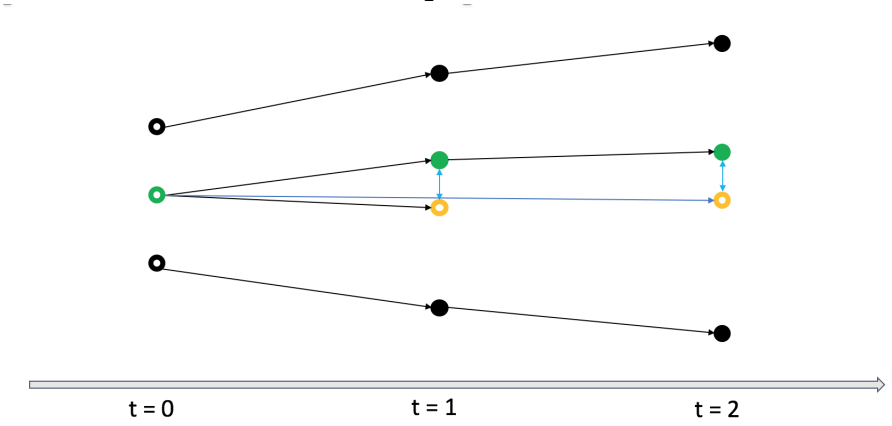
5.3 Motivation

This motivation section begins by describing a problem — local truncation error — and then a proposed solution. It then describes another problem — unbounded interpolation error in divergent areas — and another proposed solution.

Problem: The FDFP-Interpolation approach can suffer from local truncation 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 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 15a 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. [13]. 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



(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 $(\frac{1}{2}h_x^2\|f''\|)$[13]. 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 $\frac{1}{2}h_x^2\|f''\|(1 + h_tL)$ [14].



(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 15. A notional example to provide intuition of how longer basis flows can reduce error propagation.

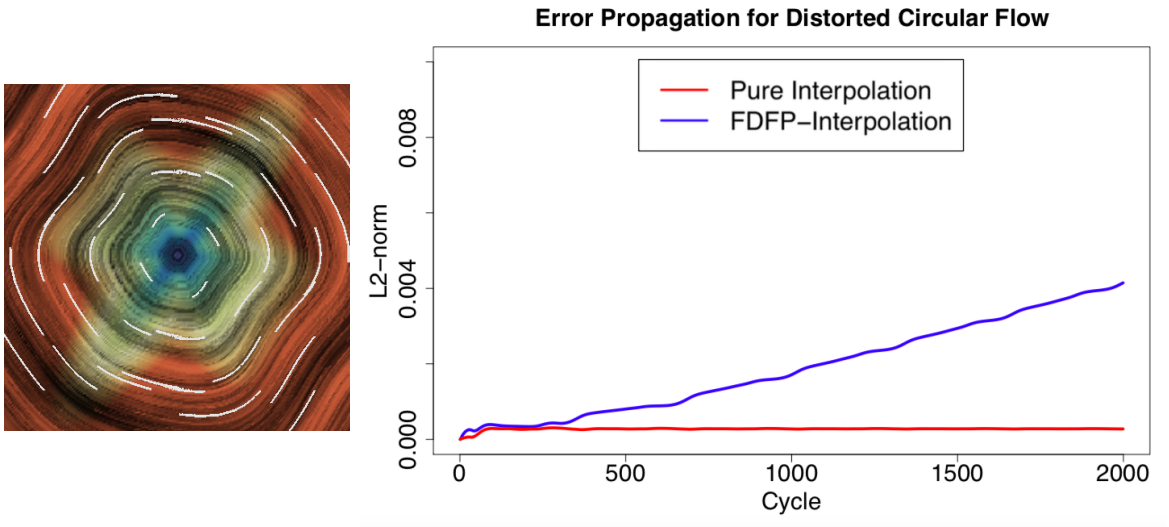


Figure 16. Motivating result comparing FDFP-Interpolation to pure interpolation on an analytic data set of distorted circular flow data. The image on the left is the LIC of the flow field (color encodes the velocity magnitude). The white lines are the FDFP basis trajectories. The image on the right is 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.

case for long simulation runs, the error propagation and accumulation can grow exponentially and lead to poor accuracy [7].

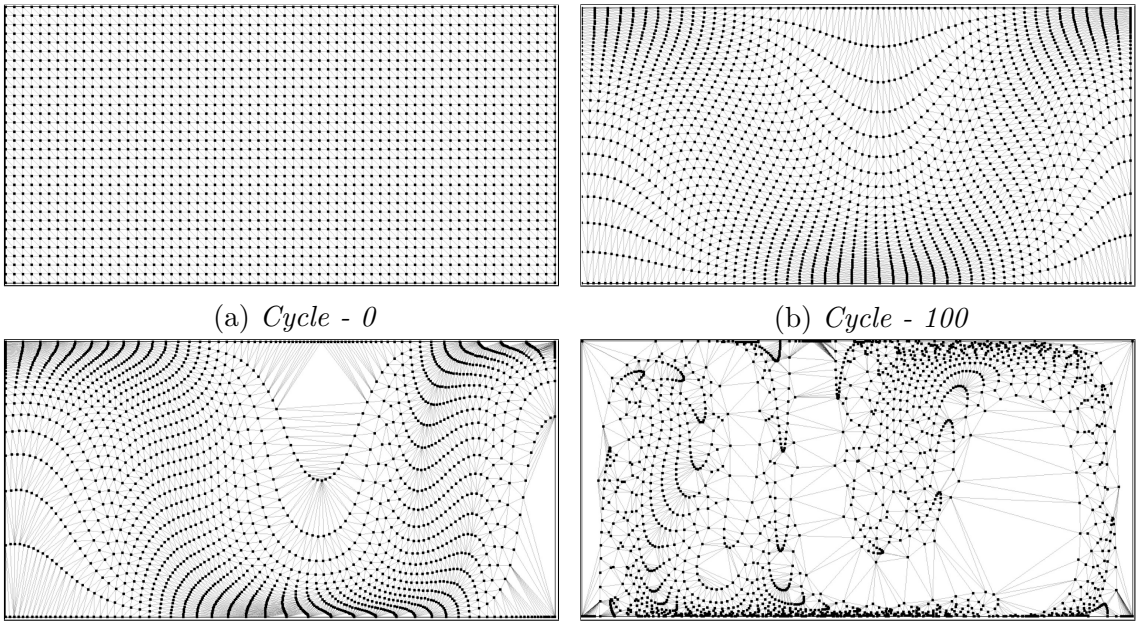
Our Solution: Extend the duration of basis flows for as long as possible. The error propagation and accumulation occurs for every instance of a stitching event (neighborhood update). We can mitigate this issue if:

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



(a) *Cycle - 0* (b) *Cycle - 100*

(c) *Cycle - 200* (d) *Cycle - 1000*

Figure 17. Particle distribution for the Double Gyre, with period set to 1000 cycles. Figure (d) shows significantly under and over represented regions of the flow.

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 [13]. Figure 15b 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 16 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,

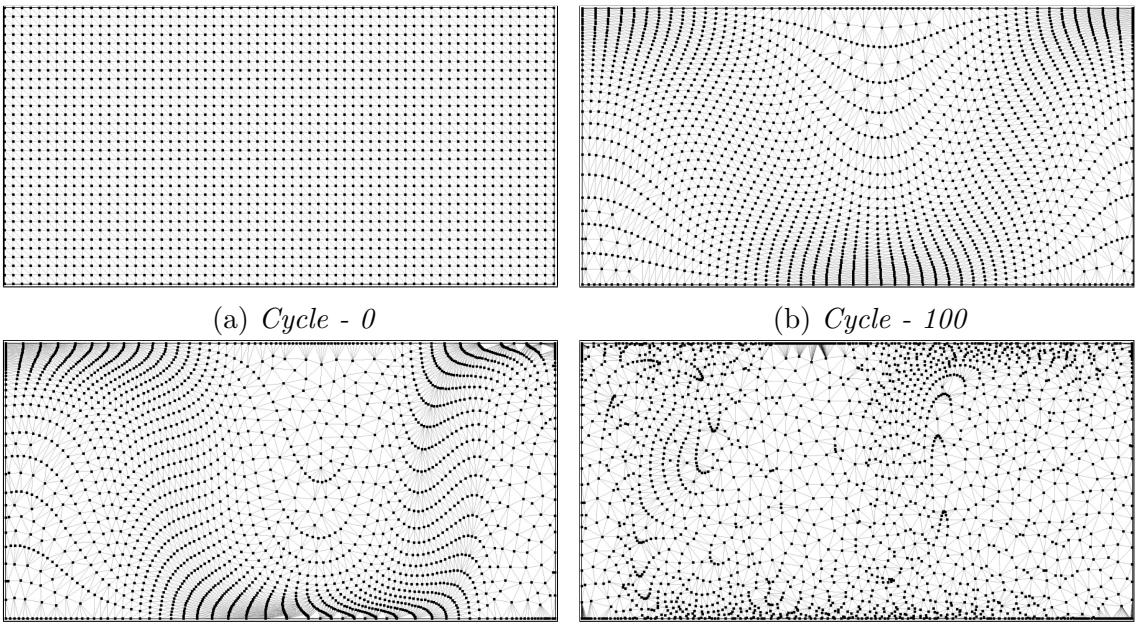
generating longer basis flows may result in certain regions having poor basis coverage, depending on the nature of the flow field. Figure 17 shows the distribution of particles at various stages when considering the Double Gyre [40]. Figure 17a shows the initial distribution of particles along a uniform grid. Figures 17b and 17c show the divergene of particles. There are observable regions in the field that are under and over represented in Figure 17d. 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. [34], 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.

5.4 VDVP-Interpolation Method



(a) *Cycle - 0* (b) *Cycle - 100*

(c) *Cycle - 200* (d) *Cycle - 1000*

Figure 18. Particle distribution after addressing under and over represented regions for the Double Gyre.

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 5.2.2. 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.

Algorithm 1: *VDVP-Interpolation Algorithm*

```
   Data: ParticleSet P, BasisFlowSet B,
   float UpperThreshold, int T_start, int T_end,
   int WriteInterval
 1 Function VDVP-Interpolation()
 2 T_current = T_start;
 3 while T_current < T_end do
 4 |   DT = Delaunay(B, T_current);
 5 |   if T_current = T_start then
 6 |   |   foreach Particle p ∈ P do
 7 |   |   |   p.NB = UpdateNBInfo(p,DT);
 8 |   |   end
 9 |   else
10 |   |   foreach Particle p ∈ P do
11 |   |   |   if EvaluateNB(p.NB, UpperThreshold) then
12 |   |   |   |   p.NB = UpdateNBInfo(p,DT);
13 |   |   |   end
14 |   |   end
15 |   end
16 |   foreach Particle p ∈ P do
17 |   |   p = Interpolate(p, p.NB);
18 |   end
19 |   T_current = T_current + WriteInterval;
20 end
```

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 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:

$$\|f(x) - Lf(x)\| \leq \frac{1}{2}R^2\|f''\|_\infty \tag{5.1}$$

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 [41].

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.

5.5 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

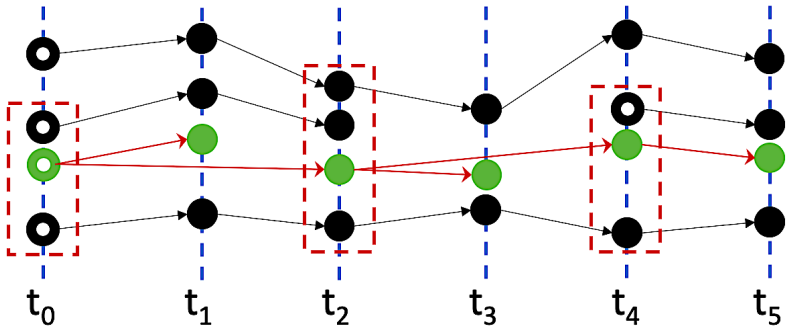


Figure 19. A notional example of VDVP-Interpolation. Basis flows are plotted in black and a sample particle trajectory being interpolated is shown in green. Hollow circles are initial positions. Red arrows show interpolation. Red dashed boxes denote neighborhood update events. t_0 - particle identifies 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).

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 17 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 18 shows a more uniform distribution of particles achieved by a flow-guided VDVP approach using particle distribution management for the Double Gyre data set.

5.6 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.

5.6.1 Configuration Parameters.

5.6.1.1 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.

5.6.1.2 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 [40]. 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 [37]. 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 [38]. The base resolution is $490 \times 490 \times 280$. A mature tornado vortex (depicted in Figure 20) 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.

5.6.1.3 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

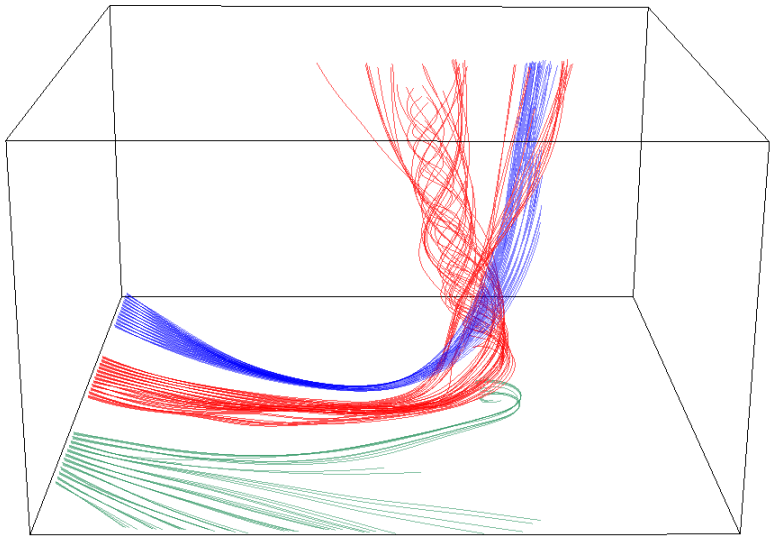


Figure 20. Pathlines traced depict a mature tornado vortex.

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:

$$UpperThreshold = CR \tag{5.2}$$

$$LowerThreshold = \frac{R}{C} \tag{5.3}$$

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.

5.6.2 Error Evaluation. We calculate particle trajectories using three methods.

– **Ground Truth** — The particle trajectory is calculated with a fourth-order Runge Kutta scheme [1] 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 20). 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 —

$$\frac{1}{p}\sum_{i=0}^{p}\frac{1}{N_C}\sum_{t=0}^{N_C}||x_{i,t}-g_{i,t}|| \tag{5.4}$$

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. [13]. For our study, we focus on the accuracy of the interpolated locations of a Lagrangian particle trajectory.

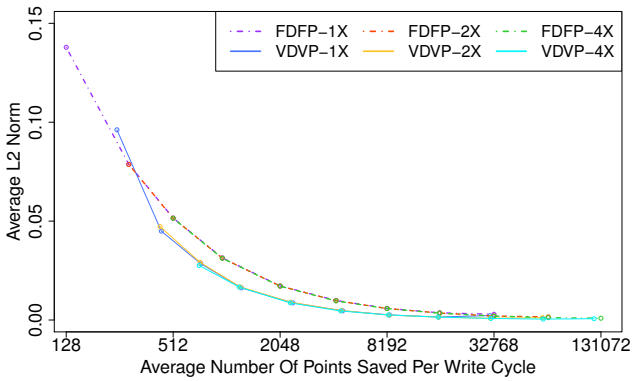
5.7 Results

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.

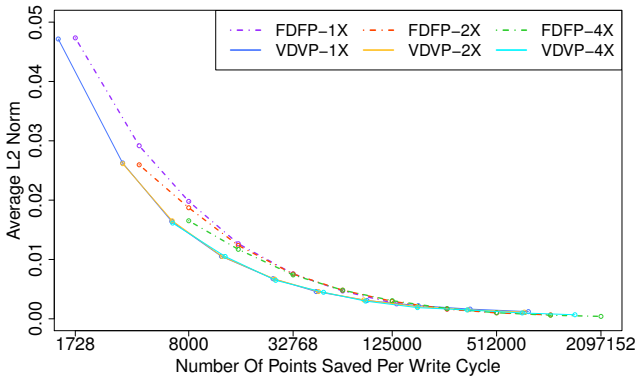
Configuration		**Double Gyre**	Configuration		**ABC**	Configuration		**Tornado**
N_C	Data	Time Per Interval	N_C	Data	Time Per Interval	N_C	Data	Time Per Interval
2048	1X	**0.1704**	2048	1X	**0.4897**	128	1X	**3.5335**
	4X	**0.1774**		4X	**0.5669**		4X	**14.6271**
128	1X	**0.1821**	128	1X	**0.6528**	32	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 3. 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.

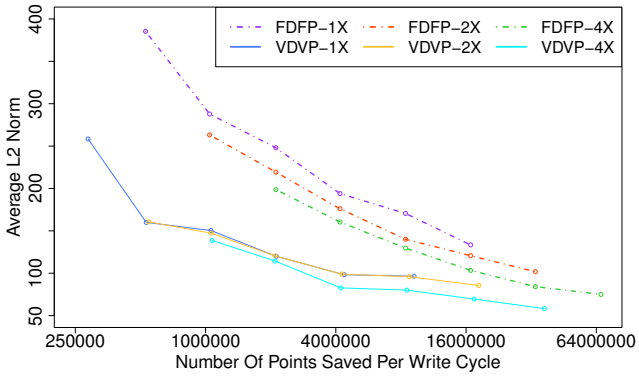
5.7.1 Accuracy and Data Storage Comparison. We analyze the results of accuracy achieved and the corresponding number of basis flows used by each approach, shown in Figure 21. The x-axis represents the average number



(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 21. Evaluation results using L2-norm. Legend indicates the configuration information.

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 21a, 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 $N_C = 128$ ($P = 8192$), 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 5.7.2.2). Considering configurations between $P_{avg} = 1372$ to $P_{avg} = 125000$ in Figure 21b, 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 21c, 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.

5.7.2 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.

5.7.2.1 Computation Time. Table 3 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 3, 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 5.8.

5.7.2.2 Neighorhood Update Rate. Figure 22 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 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

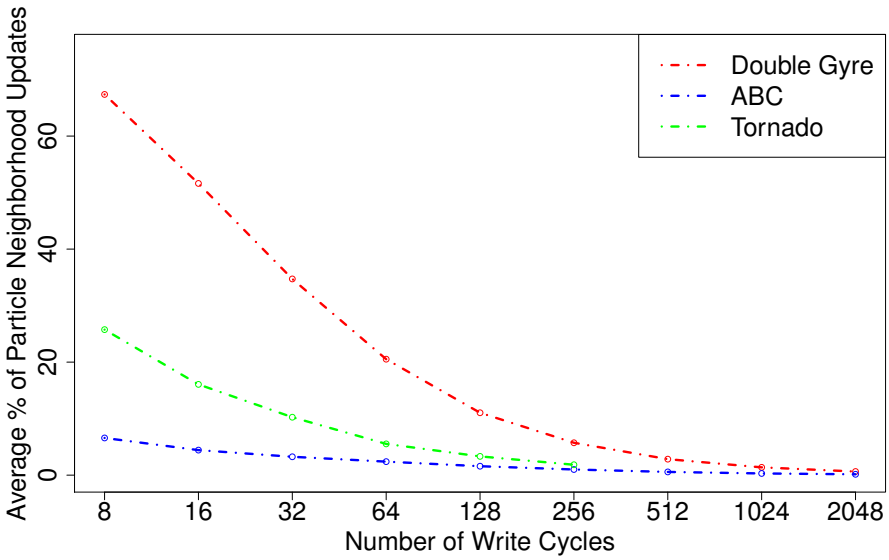


Figure 22. The average percentage neighborhood update rate over all particles for each data set. The x-axis shows the number of write cycles, i.e., the number of "opportunities" a particle being interpolated had to evaluate and decide whether to update its neighborhood. The y-axis plots the average percentage of neighborhood updates over all interpolated particle trajectories, considering all configurations, i.e., VDVP-1X, VDVP-2X, and VDVP-4X.

in the domain, we observe low values for N_{update} irrespective of the number of write cycles. This is expected given particles have rather 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.

5.8 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 **L-ISR-PHE** 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.

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.

Our study answers our research question regarding whether considering more complex seeding, termination, and interpolations methods are worthwhile — they are. Further, we feel that additional research on this topic will be fruitful (see Chapter VIII).

CHAPTER VI

SCALABLE IN SITU LAGRANGIAN FLOW MAP EXTRACTION: DEMONSTRATING THE VIABILITY OF A COMMUNICATION-FREE MODEL

Most of the text in this chapter comes from a manuscript that will be submitted for publication soon [9]. This work was a collaboration between Abhishek Yenpure, Roxana Bujack, Matthew Larsen, Kenneth Moreland, Christoph Garth, Hank Childs, and myself. I was responsible for system implementation, conducting the experiments, preparing and writing the manuscript. Abhishek Yenpure and Kenneth Moreland assisted with the generation of FTLE images. Matthew Larsen helped me with using Ascent and integrating in situ Lagrangian analysis functionality into the framework. This project was started when I was a visiting student at the Technische Universität Kaiserslautern and collaborating with Christoph Garth. Roxana Bujack advised and assisted with theoretical background in the text. Hank Childs advised, provided extensive feedback, and was involved in editing the manuscript.

6.1 Introduction

A major concern for in situ processing is the encumbrance placed on the simulation code. In situ analysis tasks are allocated a limited resource budget, and, since analysis is coupled with a simulation, analysis performance directly impacts the overall performance. In situ calculation of a Lagrangian representation, i.e., the "**ISR**" of the **L-ISR-PHE** paradigm, however, involves distributed memory particle advection to calculate sets of basis flows, introducing a heavy communication overhead. Although addressing the scalability of distributed memory integral curve computation has been extensively studied in a post hoc

setting, current techniques often do not operate within in situ constraints. Further, communicating particle information between nodes every cycle is expensive and will only become more so as supercomputers get larger. To address the challenge of in situ scalability, we evaluate the viability of a communication-free model. In the context of distributed memory integral curve computation, this is a relatively unexplored idea. However, as opposed to post hoc flow visualization (toward which most research efforts have been directed), we believe our objective of extracting a Lagrangian flow map in situ permits such a model.

In this study, we propose the Boundary Termination Optimization (BTO), a simple, yet novel, communication-free algorithm to improve the performance of in situ *extraction* of Lagrangian flow maps. We demonstrate the viability and limitations of this method by considering multiple simulation data sets and parameter configurations. Our study finds that a communication-free model, for the several practical configurations considered, achieves 3x speed-up while calculating flow maps in situ and can approximate the complete flow map while maintaining over 96% reconstruction accuracy in several cases compared to using communication during the first phase of **L-ISR-PHE**.

6.2 Background and Related Work

A key operation for calculating a Lagrangian flow map in situ is computing pathlines in a distributed memory environment. Efficient computation of integral curves in a distributed memory environment is an extensively researched field. However, the majority of works are limited to steady state vector fields in a post hoc environment. The primary challenge addressed by these works is improving the scalability, load balance, and overall efficiency of distributed memory integral curve computation. Typical parallelization strategies adopted to improve performance are

parallelize-over-data [42, 43, 44, 45, 46], parallelize-over-particles [47, 48, 49, 50], or a hybrid approach [51, 52, 53]. Parallelize-over-data techniques determine a domain decomposition and assign a subset of the total domain to each node. While calculating integral curves, these methods communicate particles between processors when required. Parallelize-over-particles techniques assign a set of particles to each node and load data from disk on demand. For a complete review of the algorithms, we refer the reader to a recent survey by Zhang et al. [54].

Although aiming to improve performance for different flow visualization tasks and involving similar keywords as our work work, the following related works are only applicable to steady state vector fields in a post hoc setting. Bleile et al. [55] accelerated streamline calculation by swapping traditional Eulerian and Lagrangian-based advection at node boundaries. In this case, after a particle is communicated across a boundary, a previously computed mapping is used to transport the particle across the entire node. Liao et al. [56] presented a communication-free 3D LIC technique. They limit communication by using a preprocessing step to regroup unstructured grid cells and restricting particle advection to within the confines of a single cell.

Existing algorithms are difficult to adopt in the context of in situ calculation of pathlines. Operating in an in situ environment introduces new constraints with regard to which process can access what domain subset in space and time. First, domain decomposition and distribution are simulation-determined. Second, the time-varying nature of simulation data in conjunction with in situ memory constraints means only a single time step can be accessed at any given time, and communicating this information across processors each step is prohibitively expensive. These constraints complicate techniques like data prefetching,

rearrangement, or completing particle advection within a node before particle exchange. (In situ methods are currently limited to advancing a particle by a single step before the vector field changes.) Further, the problem of poor scalability remains when considering a large number of processors and communication between them every cycle.

6.3 Boundary Termination Optimization

In this section, we first define requirements for extracting flow field information as a set of pathlines in situ. Next, we describe Boundary Termination Optimization (BTO), which results in the communication-free technique Lagrangian-BTO. Our demonstration of the viability of a communication-free model is the main contribution of this study. Further, we discuss the accuracy-performance tradeoff when comparing a communication-based versus communication-free model. Finally, we provide the details of the Lagrangian-based advection scheme we use for reconstruction and a theoretical error analysis.

The requirements surrounding the calculation of integral curves in distributed memory depends on whether the computation is for the purpose of post hoc visualization and analysis or in situ extraction of a flow map. Whereas in a post hoc setting an integral curve must continue particle integration across node boundaries, this is not necessary to calculate a Lagrangian representation of the flow field. We identify different requirements when extracting a Lagrangian flow map in situ and define them as follows:

1. Extraction of a flow map or set of pathlines in situ should demonstrate good scalability.
2. Flow field reconstruction using extracted pathlines should be accurate.

The method by Agranovsky et al. [5] demonstrated only the second requirement. We implement the method by Agranovsky et al. [5] using MPI [57] for communication and refer to this technique as Lagrangian-MPI. In this paper, we use Lagrangian-MPI as a baseline for comparison.

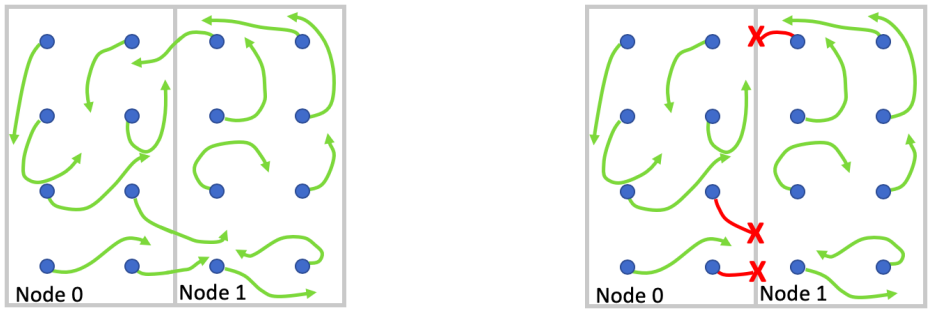
6.3.1 In Situ Extraction Using BTO. Our contribution with this work is a simple communication-free algorithm for extracting a Lagrangian flow map in situ. The benefit of this approach is that it has less execution time and improved scalability characteristics, which reduces the burden on the simulation code. To improve performance, our approach requires a small modification to Lagrangian-MPI: eliminate information exchange and synchronization.

Similar to Lagrangian-MPI, we use an initially uniform seed placement and advect particles for predetermined nonoverlapping intervals of time. However, as opposed to continuing particle integration across node boundaries, our approach terminates and discards these particle trajectories. Figures 23a and 23b illustrate notional examples of basis flows calculated by Lagrangian-MPI and Lagrangian-BTO, respectively. Thus, we store only those particle trajectories that remain within the domain until the end of the interval. Terminating particles that require communication across node boundaries to continue trajectory integration, allows the approach to remain communication-free. Since processors do not exchange particles, the Lagrangian analysis operator on each processor can operate independently and asynchronously.

We build both Lagrangian analysis techniques, i.e., Lagrangian-MPI and Lagrangian-BTO, as in situ analysis filters using the VTK-m [58], VTK-h [59], and Ascent [59] libraries. VTK-m is a platform portable scientific visualization library for shared memory parallel environments. VTK-h is a distributed memory wrapper

around VTK-m. Ascent is an in situ visualization infrastructure that we use to both integrate with simulations and create a workflow when loading data sets from disk.

The Lagrangian-BTO filter has two operations to perform: particle advection and particle management. Particle advection is performed using RK4 interpolation implemented as a VTK-m worklet [60]. Particle management involves tracking particle trajectories, evaluating the validity, and managing memory to prevent invalid particles from being launched on GPU threads during advection. The Lagrangian-MPI filter has to perform three operations: particle advection, particle management, and communication. Consistent with the Agranovsky approach, communication of particles and particle information between ranks is performed using asynchronous, non-blocking, buffered MPI communication. Additionally, particle management further includes tracking of *internal* and *external* particles in order to return particles to originating nodes at write cycles.



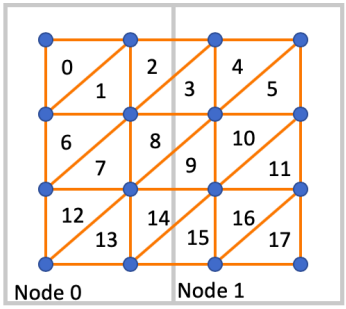
(a) Lagrangian-MPI (b) Lagrangian-BTO

Figure 23. Notional examples of calculating basis flows. Only green trajectories are stored to disk.

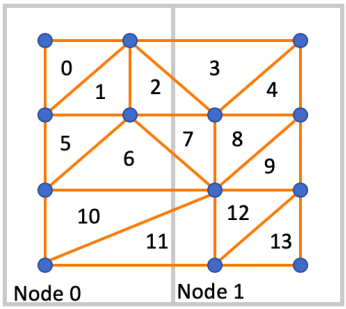
Accuracy-Performance Tradeoff — The loss of information in the form of terminated and discarded particle trajectories reduces the quality of flow

reconstruction. Flow information that will typically be lost near boundaries can, however, be interpolated using additional information from adjacent processes post hoc. With respect to simulation overhead, the communication-free model offers a reduced cost and improved scalability. We believe evaluating the accuracy-performance tradeoff and determining the viability of a communication-free model as an alternative low-cost choice for a scientist or researcher is valuable. Our hypothesis regarding this method is that the execution time will be substantially improved (since there is no communication required), but the accuracy will be only modestly affected.

Finally, the Agranovsky work demonstrated that the Lagrangian method can be much more accurate than the Eulerian approach, including some cases where the accuracy improved by over 10x. Even though our practice of terminating particles at the boundary will reduce accuracy compared to the Agranovsky approach, we still would be much more accurate than the Eulerian approach. If we also can demonstrate significantly faster execution times, then we believe our proposed method would be appealing to future researchers and domain scientists in many settings.



(a) Neighborhoods using 23a



(b) Neighborhoods using 23b

Figure 24. Notional examples of neighborhoods produced using triangulation over basis flows shown in Figure 23. The blue circles represent starting locations of basis flows saved to disk.

6.3.2 Post Hoc Reconstruction. To reconstruct the complete flow map or to trace new pathlines, we set up a parallelize-over-data distributed memory Lagrangian-based advection scheme that is conceptually similar to previous work [5, 21, 7]. For a given interval, each process loads basis flows generated from its own and adjacent processes for that interval. For any new trajectory to be calculated using basis flows, the particle must identify a neighborhood (convex hull) of basis flows to follow for the duration of an interval. We perform a Delaunay triangulation over the start locations of loaded basis flows to identify neighborhoods for tracing new particle trajectories. Figure 24 illustrates a notional example of triangulation using basis flows from the two nodes demonstrated in Figure 23. Using basis flows data from adjacent neighboring processes, i.e., flow information from all directions, allows approximation of regions where information has been lost. Once a particle neighborhood is identified, the location of the particle at the end of the interval can be calculated by following the neighborhood of basis flows. In our work, we use Barycentric coordinate interpolation to calculate the particle's end position. To calculate a pathline for a duration greater than the interval of basis flows, a trajectory can be stitched together by using basis flows of successive nonoverlapping intervals.

We implemented our reconstruction workflow using CGAL for 3D parallel Delaunay triangulation [61] and the vtkProbeFilter from the VTK library [62] for Barycentric coordinate interpolation. Further, we performed all our reconstruction and accuracy measurements on our in-house research cluster.

6.3.3 Theoretical Error Analysis. We use a one-dimensional linear interpolation L of a function $f : \mathbb{R} \to \mathbb{R}$ for $x \in [x_0, x_1] \subset \mathbb{R}$

$$L_{f(x_0),f(x_1)}(x) = \frac{x - x_0}{x_1 - x_0} f(x_1) + \frac{x_1 - x}{x_1 - x_0} f(x_0). \tag{6.1}$$

The higher dimensional result satisfies Equation (6.1) for each component.

In our approach, a particle starting at x_1 that reaches the node boundary during the interval of advection is terminated. Thus, its function value (the flow map) is not known. However, we can reconstruct it from its known neighbors $L_{f(x_0),f(x_2)}(x_1)$. Consider a particle $x \in [x_0, x_1] \subset \mathbb{R}$ whose path is interpolated post hoc using this reconstructed value. We get the same result as if we had used the closest existing neighbors directly, because let L' denote $L_{f(x_0),L_{f(x_0),f(x_2)}(x_1)}(x)$, then

$$\begin{aligned}
L' &\overset{(6.1)}{=} \frac{x-x_0}{x_1-x_0} L_{f(x_0),f(x_2)}(x_1) + \frac{x_1-x}{x_1-x_0} f(x_0) \\
&\overset{(6.1)}{=} \frac{x-x_0}{x_1-x_0}\left(\frac{x_1-x_0}{x_2-x_0} f(x_2) + \frac{x_2-x_1}{x_2-x_0} f(x_0)\right) + \frac{x_1-x}{x_1-x_0} f(x_0) \\
&= \frac{x-x_0}{x_2-x_0} f(x_2) + \frac{(x_2-x_1)(x-x_0) + (x_2-x_0)(x_1-x)}{(x_2-x_0)(x_1-x_0)} f(x_0) \\
&= \frac{x-x_0}{x_2-x_0} f(x_2) + \frac{x_2-x}{x_2-x_0} f(x_0) \\
&\overset{(6.1)}{=} L_{f(x_0),f(x_2)}(x).
\end{aligned} \tag{6.2}$$

Bujack et al. [13] previously established that the post hoc interpolation of pathlines is a numerical one-step integration method [63]. Its accuracy is bounded by its global truncation error at stitching step $n \in \mathbb{N}$ of

$$e_n \leq \frac{d^2}{8}(t_n - t_0) h_x^2 \max_{\tau \in [t_0,t_n]} \max_{\zeta \in \mathbb{R}^d} \|H_{\dot{F}^\tau_{t_{j-1}}}(\zeta)\|_\infty e^{L(t_n - t_0)} \tag{6.3}$$

with dimension d, start time t_0, end time t_n, spatial Lipschitz constant L, Hessian H of the temporal derivative of the flow map $\dot{F}$, and spatial distance h_x between the basis flows. As a result, the interpolation error is $O(h_x^2)$ if the flow map has bounded second derivatives in space and first derivatives in time, which is a reasonable assumption for a differentiable vector field, because the solutions of an initial value problem depend smoothly on the initial conditions and time [64].

Equation (6.2) shows that the error bound $O(\tilde{h}_x^2)$ still holds but with a larger $\tilde{h}_x > h_x$. Its size is determined by the size of missing information, which is limited by the maximum distance particles can move in one time interval. If a particle is seeded further than $\max_{x_i \in \mathbb{R}^d} \max_{j=1..n} \|F_{j-1}^j(x_i)\|$ away from the

node boundary, it cannot reach it and must therefore have the correct flow map information. In the worst case, we can have missing information on both sides of the boundary, and therefore we get

$$\begin{aligned} \tilde{h}_x &\leq 2h_x + 2 \max_{x_i \in \mathbb{R}^d} \max_{j=1..n} \|F_{j-1}^{j}(x_i)\| \\ &\leq 2h_x + 2h_t \max_{x \in \mathbb{R}^d} \max_{t \in [t_{j-1}, t_j]} \|v(x,t)\| \end{aligned} \tag{6.4}$$

with the underlying velocity field $v : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$ and the temporal step size h_t, which is the interval time between storing data to disk, which is usually around one-thousandth of the total integration time. Please note that the future increase of the global truncation error of a particle that traverses this region can continue even after it has left the region, but for all particles that never enter the region close to the boundary, the original error bound holds.

6.4 Study Overview

In this section, we describe our experiments to evaluate performance and reconstruction accuracy, and the metrics we use to measure the same.

6.4.1 Experiment Setup. To evaluate the viability of the Lagrangian-BTO analysis filters, we set up two workflows.

- **WF1:** In situ weak scaling study to evaluate speed-up.
- **WF2:** Evaluation of reconstruction accuracy by varying parameters used in flow map generation.

For the **WF1** workflow, Ascent is directly connected to a simulation code. Lagrangian analysis parameters are specified as input to the Ascent pipeline. The simulation code generates vector field data and pauses when it invokes Ascent calls in order to perform in situ analysis. We scaled the number of processors used and proportionately increased the size of the simulation grid. Our **WF1** experiments provide insight into the performance of both Lagrangian analysis filters at scale.

For the **WF2** workflow, we load data set files from disk to create a theoretical in situ setup, and then call Ascent to initiate the Lagrangian analysis filters. We consider a fixed number of MPI tasks and nodes for **WF2**, i.e., we use a fixed domain decomposition for all tests. Our **WF2** experiment configurations are designed to understand the range of specifications under which Lagrangian-BTO extracts accurate flow maps and to identify the potential limitations of using this method. We vary the value of the interval parameter (i.e., the number of cycles we wait before writing to disk) to understand the effect of longer advection intervals on flow map generation using the Lagrangian-BTO approach. In general, the longer the interval, the greater the probability of particles reaching the boundary and being terminated. Further, we consider the impact of various data reduction options. When using a very sparse number of particles to capture the behavior of the flow, the effect of losing particles to boundary termination could be greater. Similar to the format used in other chapters, data reduction values are described in the form of 1:X, i.e., one particle for every X grid points.

6.4.2 Performance Metric. We measure performance in terms of the execution time of the Lagrangian analysis filters. All of our timings are measured using the timing functionality in Ascent. We measure the average time per cycle, which includes time to perform particle advection, particle management, and also communication in the case of Lagrangian-MPI. To simplify understanding the scalability of both Lagrangian analysis filters with respect to communication costs, we exclude cycles at the end of an interval. These cycles include a communication cost incurred to return all particles to the respective origin nodes for Lagrangian-MPI and an I/O cost to write information to disk for both methods. In this

paper, we do not analyze the parallel I/O times, and we consider I/O optimization methods to be beyond the scope of this work.

6.4.3 Accuracy Metric. We measure the accuracy of flow field reconstruction by the Lagrangian-BTO analysis filter relative to the accuracy achieved by the corresponding Lagrangian-MPI analysis filter. Comparisons of Lagrangian-MPI to the traditional Eulerian method can be found in previous works [5, 7]. We measure total flow volume error using the average L2-norm over all samples considered. The total average L2-norm is calculated as

$$\frac{1}{p}\sum_{i=0}^{p}||b_{i,t} - m_{i,t}|| \tag{6.5}$$

where p is the total number of particles, $b_{i,t}$ is the location of a Lagrangian-BTO interpolated particle i at time t, and $m_{i,t}$ is the location of the Lagrangian-MPI particle i at time t.

For a given total average L2-norm value $\mathbf{L}$, the reconstruction accuracy percentage is proportional to the length of cell side $\mathbf{C}$ for that specific configuration, and is calculated as

$$Accuracy\% = \frac{C - L}{C} \times 100 \tag{6.6}$$

We note that we use the total average L2-norm in two contexts. First, to measure error when reconstructing the complete flow map as generated by the Lagrangian-MPI method. Second, to measure error of new pathlines traced using basis flows generated by both methods when compared to a ground truth.

In addition to the above total flow volume error measure, we report the maximum L2-norm in two forms — the greatest maximum L2-norm across all

interval reconstructions, i.e., the error of the least accurately interpolated single basis flow, and the average maximum L2-norm across all intervals.

6.4.4 Data Sets. We consider four data sets: a Cloverleaf3D simulation, Arnold-Beltrami-Childress flow, a Jet flow simulation, and a Nyx cosmology simulation.

6.4.5 Runtime Environment. We tested the Lagrangian analysis techniques by running our experiments on Summit, a supercomputer at ORNL. Each node of Summit has two IBM Power9 CPUs, each with 22 cores running at 3.8 GHz and 512 GBytes of DDR4 memory. Further, nodes on Summit have enhanced on-chip acceleration with each CPU connected via NVLink to 3 GPUs, for a total of 6 GPUs per node. Each GPU is an NVIDIA Tesla V100 with 5120 CUDA cores, 6.1 TeraFLOPS of double precision performance, and 16 GBytes of HBM2 memory.

6.5 Results

We organize our results into four subsections, 6.5.1 to 6.5.4. Each subsection is focused on one data set. Specifically, in subsection 6.5.1, we consider the Cloverleaf3D simulation with workflow **WF1** for our weak scaling study and workflow **WF2** for our strong scaling study. In subsections 6.5.2, 6.5.3, and 6.5.4, we consider the ABC, Nyx, and Jet flow simulations, respectively, and run experiments with workflow **WF2**.

For all data sets, we measured the accuracy of basis flows generated for every interval across all nodes. Tables 4, 5, 6, and 7 show reconstruction accuracy averaged over all the intervals, i.e., average reconstruction accuracy for the entire simulation duration. Figure 31 shows the change in reconstruction accuracy over every interval and the correlation to the number of particles terminated during

that interval for a single configuration of each data set. Figures 31a, 31c, 31e, and 31g use bubble size to represent the number of particles terminated and the corresponding figures 31b, 31d, 31f, and 31h show average L2-norm curves and reconstruction accuracy as a percentage.

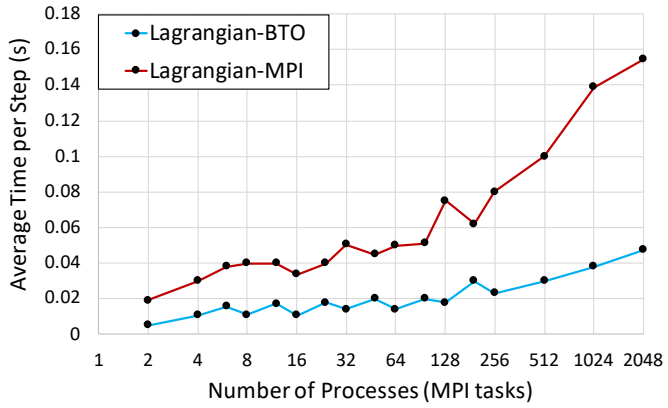
To supplement our quantitative evaluation, Figures 26a, 26b, 28, 29, and 30 provide a qualitative comparison using colormapped visualizations of surfaces of subvolumes of FTLE scalar fields generated post hoc using basis flows calculated by Lagrangian-MPI and Lagrangian-BTO. Well-defined ridges in the FTLE field (identified by high scalar values), used to visualize Lagrangian Coherent Structures, are of particular interest in these figures.

Nodes	MPI Ranks	GPUs/ Node	Dims	Step Size	L-BTO (s)	L-MPI (s)	Speed-up	Discarded %	Greatest Max L2-norm	Avg. Max L2-norm	Total Avg. L2-norm	Accuracy %
1	2	2	81^3	0.038	0.0050	0.0190	**3.8x**	1.9	2.52×10^{-3}	7.58×10^{-4}	4.53×10^{-4}	**99.6**
1	4	4	102^3	0.029	0.0106	0.0301	**2.8x**	2	2.44×10^{-3}	7.68×10^{-4}	4.19×10^{-4}	**99.5**
1	6	6	116^3	0.025	0.0158	0.0380	**2.4x**	2.8	3.43×10^{-3}	1.20×10^{-3}	4.89×10^{-4}	**99.4**
2	8	4	128^3	0.023	0.0109	0.0405	**3.7x**	1.8	1.61×10^{-3}	6.49×10^{-4}	2.60×10^{-4}	**99.6**
2	12	6	146^3	0.019	0.0173	0.0398	**2.3x**	2.4	2.84×10^{-3}	1×10^{-3}	3.09×10^{-4}	**99.5**
4	16	4	161^3	0.017	0.0107	0.0338	**3.1x**	2.9	1.11×10^{-2}	1.97×10^{-3}	4.38×10^{-4}	**99.2**
4	24	6	184^3	0.015	0.0178	0.0410	**2.3x**	4.6	5.83×10^{-3}	2.35×10^{-3}	6.28×10^{-4}	**98.8**
8	32	4	203^3	0.013	0.0140	0.0506	**3.6x**	4.1	3.11×10^{-3}	1.44×10^{-3}	3.83×10^{-4}	**99.2**
8	48	6	232^3	0.011	0.0201	0.0449	**2.2x**	4.9	5.70×10^{-3}	3.15×10^{-3}	4.45×10^{-4}	**98.9**
16	64	4	256^3	0.010	0.0140	0.0504	**3.6x**	4.5	8.39×10^{-3}	3.58×10^{-3}	3.62×10^{-4}	**99.1**
16	96	6	293^3	0.009	0.0200	0.0510	**2.5x**	—	—	—	—	—
32	128	4	322^3	0.008	0.0180	0.0750	**4.1x**	—	—	—	—	—
32	192	6	370^3	0.007	0.0301	0.0620	**2.0x**	—	—	—	—	—
64	256	4	406^3	0.006	0.0230	0.0808	**3.5x**	—	—	—	—	—
128	512	4	512^3	0.005	0.0303	0.1001	**3.3x**	—	—	—	—	—
256	1024	4	645^3	0.004	0.0380	0.1390	**3.6x**	—	—	—	—	—
512	2048	4	812^3	0.003	0.0475	0.1544	**3.2x**	—	—	—	—	—

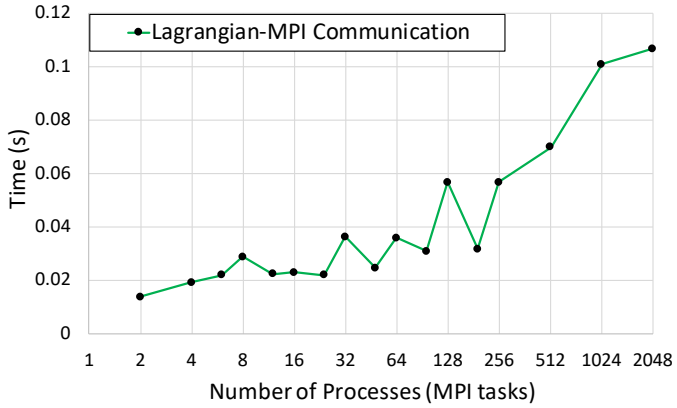
Table 4. Weak scaling configurations and timing results for the Cloverleaf3D data set **WF1** experiments. Lagrangian-BTO and Lagrangian-MPI columns show the average time per step in seconds. We post-processed and measured accuracy for only a subset of the **WF1** experiments (10 of 17), since calculating the reconstruction accuracy takes prohibitively long periods of time on our local cluster. Each node operates on an approximately 64^3 grid. Reported results are measured across all intervals.

6.5.1 Cloverleaf3D Simulation. Cloverleaf3D is a three-dimensional version of the Lagrangian-Eulerian explicit hydrodynamics mini-application

Cloverleaf [65]. It has been developed and used to evaluate techniques targeting Exascale applications.



(a)

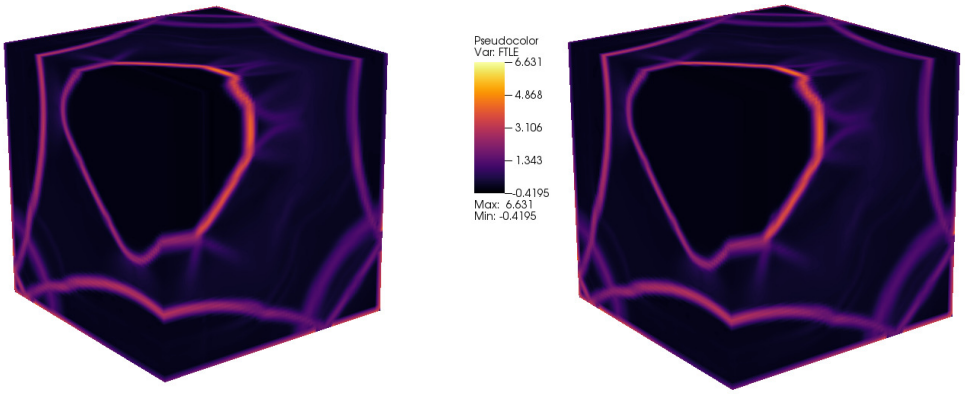


(b)

Figure 25. Weak scaling study results for the Cloverleaf3D data set. In the top sub-figure 25a, each curve plots the average time required for a single step for an increasing number of MPI tasks. Lagrangian-BTO is on average 3x faster than Lagrangian-MPI. The bottom sub-figure 25b shows the approximate time required for MPI communication by the Lagrangian-MPI analysis filter averaged across all processes. All measurements are in seconds.

6.5.1.1 Weak Scaling.

We performed **WF1** experiments, i.e., an in situ weak scaling study to evaluate performance, using the Cloverleaf3D simulation on Summit. For each run, we terminated the simulation after 500 cycles. Given



(a) Lagrangian-MPI

(b) Lagrangian-BTO

Figure 26. 26a and 26b compare FTLE visualizations generated post hoc using basis flows from Lagrangian-MPI and Lagrangian-BTO, respectively.

that our study varies the resolution of the simulation with the number of processes, the total number of simulation steps varies. 4 of 17 experiment configurations completed before 500 cycles and all the simulations that are terminated at 500 cycles reach different stages of the simulation (variable step size). In general, the greater the spatial resolution of the data set, the greater the number of cycles required to reach completion. Our experiment configurations span 81^3 across 2 MPI tasks to 812^3 across 2048 MPI tasks, with each MPI rank operating on an approximately 64^3 grid. In each case, a single MPI task is allocated a single GPU. Thus, our smallest configuration used 2 GPUs on a single compute node, and the largest used 2048 GPUs across 512 compute nodes. Additionally, given that each node on Summit has 6 GPUs, we varied the number of GPUs utilized on a single node to gauge on-node MPI communication optimizations and performance of particle advection using multiple GPUs on the same node. For each experiment, we set the maximum step size to 0.1. However, Cloverleaf3D uses an adaptive step size based on the simulation grid resolution. We report the average step size taken by the simulation, configuration information (number of nodes, MPI ranks, and GPUs

per node, and dimensions), and scalability performance results in Table 4. Further, each weak scaling test extracted basis flows at an interval of every 25 cycles and used a data reduction of 1:8, i.e., one particle for every eight grid points.

Performance — Figure 25a compares the average time required per step by each technique. Lagrangian-BTO scales better than Lagrangian-MPI because it is a communication-free analysis filter. For the extraction of Lagrangian flow maps in situ, Lagrangian-BTO demonstrates an average of 3x speed-up over Lagrangian-MPI. We observe the cost of particle advection (for both techniques) increases as the scale of the simulation increases. However, each process operates on an approximately 64^3 grid irrespective of the total number of MPI tasks, and each Lagrangian-BTO process operates without any knowledge of other processes. In addition to the number of particles being advected, multiple variables influence the cost of the particle advection worklet, namely, cell size, step size, and the vector field. Performing Runge Kutta fourth-order interpolation could require interpolating velocity information from multiple cells (determined by cell size, step size, and the vector field). Although the increased cost of particle advection affects the time required by both Lagrangian analysis techniques, exact comparisons between different configurations (i.e., rows in Table 4) would not account for the above parameters. We believe the performance of the particle advection worklet considering the above parameters is worth future investigation. Further, use of a faster particle advection kernel would result in greater speed-ups for the Lagrangian-BTO technique.

Varying number of GPUs/node — The "sawtooth" nature of the plots in Figures 25a and 25b is a result of alternating the number of GPUs being utilized between 4 and 6. Particle advection performs better with 4 GPUs per node versus

6. The use of shared memory by multiple GPUs on a single node and saturation of the NVLink by the VTK-m particle advection worklet causes this effect. Figure 25b captures the difference in time between both curves, i.e., approximates the time spent on communication, and shows a reduction in MPI communication costs when using 6 ranks versus 4 per node, albeit scaling poorly as the number of nodes increases. On-node MPI communication optimizations contribute to better performance when grouping a larger number of MPI tasks on each node. However, as the number of nodes increases, the cost of inter-node communication remains high in comparison to on-node communication.

Accuracy — Higher resolution simulation configurations use a proportionately larger number of particles and thus generate more basis flows. We measured accuracy for only a subset of the **WF1** experiments (10 of 17), since calculating the reconstruction accuracy takes prohibitively long periods of time on our local cluster. Lagrangian-BTO terminated between 2-5% of basis flows on average across all experiments, i.e., 2-5% less data was saved to disk. However, after we interpolate the missing trajectories using the saved basis flows, we approximate the complete flow map (as generated by Lagrangian-MPI) with over 99% accuracy on average using the L2-norm metric.

Figures 26a and 26b compare FTLE visualizations generated using approximately 2M basis flows saved by each technique (configuration in row 10 of Table 4). We observe no significant differences in the visualizations with both methods capturing the FTLE ridges with the same quality.

Overall, our weak scaling study showed that for the Cloverleaf3D data set, Lagrangian-BTO was on average 3x faster, while generating 99% as accurate flow maps on average and qualitatively comparable post hoc FTLE visualizations as

Lagrangian-MPI. In situ analysis using Lagrangian-MPI contributed between 5-12% of the total simulation time, and in most cases, Lagrangian-BTO required 50%-80% less time than the corresponding Lagrangian-MPI configuration.

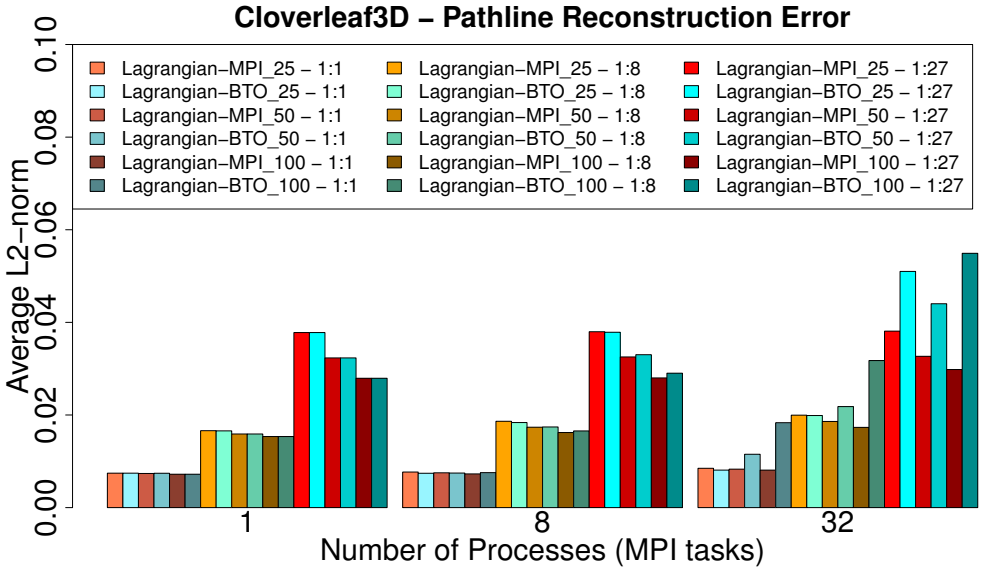


Figure 27. Strong scaling pathline reconstruction error plot for the Cloverleaf3D data set **WF2** experiments. We use warm colored bars for Lagrangian-MPI and cool colored bars for Lagrangian-BTO. Bars are grouped from left to right by number of MPI tasks used, i.e., in increasing order of degree of domain decomposition. Within each group of bars, configurations are subgrouped by data reduction options, i.e, 1:1, 1:8, and 1:27. Within a subgroup, pairs of matching Lagrangian-MPI and Lagrangian-BTO tests are ordered from left to right by interval value. In these experiments, a 64^3 dimension data set was used. The total number of simulation cycles was set to 800. We traced pathlines for 800 cycles using the basis flows generated by each Lagrangian analysis technique and compared them to the ground truth.

6.5.1.2 Strong Scaling.

Whereas the weak scaling study demonstrated the viability of using Lagrangian-BTO, the strong scaling study helps us understand its limitations. We considered a 64^3 grid and decomposed it into smaller grids based on the number of MPI tasks. Further, for these **WF2** experiments, we considered interval values of 25, 50, and 100, and data reduction values of 1:1, 1:8, and 1:27. The interval determines how long particles advect

before their end locations are saved. The longer the interval, the greater chance of particles exiting the node domain and vice versa. Further, the number of particles that exit the node domain and terminate directly impacts the accuracy of the approximation. Figure 27 shows the accuracy results when decomposing the domain across 1, 8, and 32 MPI tasks.

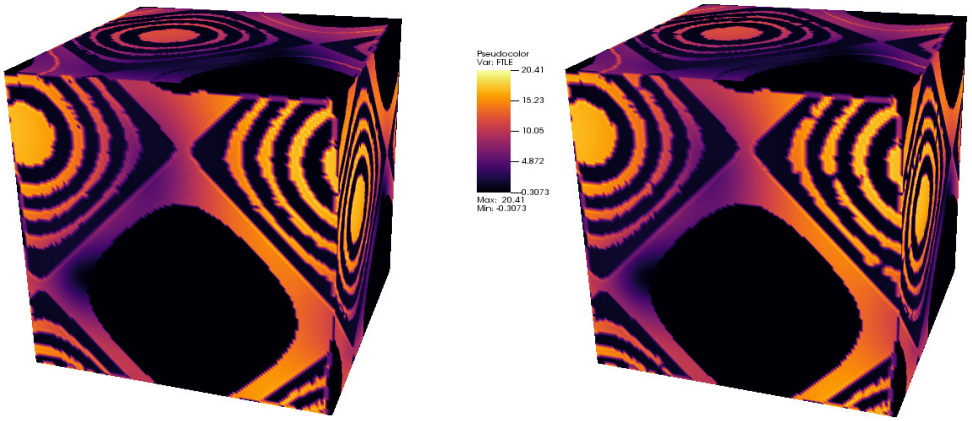
Interval	Reduction	L-BTO (s)	L-MPI (s)	Speed-up	Discarded %	Greatest Max L2-norm	Avg Max L2-norm	Total Avg L2-norm	Accuracy %
25	1:1	0.0059	0.0143	**2.3x**	2.7	6.85×10^{-4}	4.97×10^{-4}	6.15×10^{-5}	**99.7**
50	1:1	0.0066	0.0145	**2.1x**	5.5	3.03×10^{-3}	1.85×10^{-3}	2.31×10^{-4}	**99.0**
100	1:1	0.0067	0.0144	**2.1x**	10.9	7.87×10^{-3}	6.81×10^{-3}	1.09×10^{-3}	**95.5**
25	1:8	0.0026	0.0065	**2.4x**	2.6	1.40×10^{-3}	3.41×10^{-4}	3.93×10^{-5}	**99.8**
50	1:8	0.0033	0.0086	**2.6x**	5.4	3.71×10^{-3}	1.23×10^{-3}	1.32×10^{-4}	**99.4**
100	1:8	0.0026	0.0075	**2.8x**	10.8	6.15×10^{-3}	2.63×10^{-3}	3.65×10^{-4}	**98.5**
25	1:27	0.0024	0.0065	**2.7x**	1.6	5.24×10^{-5}	4.87×10^{-5}	1.61×10^{-5}	**99.9**
50	1:27	0.0029	0.0048	**1.6x**	5.1	2.62×10^{-3}	1.28×10^{-3}	1.23×10^{-4}	**99.5**
100	1:27	0.0027	0.0053	**1.9x**	10.4	1.25×10^{-2}	6.53×10^{-3}	5.01×10^{-4}	**97.9**

Table 5. Speed-up and reconstruction accuracy results for the ABC data set **WF2** experiments. Lagrangian-BTO and Lagrangian-MPI columns show average time per step in seconds. Reported results are measured across all intervals.

For these experiments, we measured accuracy by calculating pathlines using the basis flows generated by each technique and evaluating their accuracy in comparison to a ground truth at interpolated locations. The ground truth is calculated by performing RK4 advection using the full spatial and temporal resolution of the data set. We placed 1,000 particles in the domain to generate pathlines for this evaluation. We compared the similarity between ground truth and pathlines generated by interpolating Lagrangian basis flows using the average L2-norm metric. Our metric compared the accuracy of interpolated points along the trajectory [8]. Lastly, for these experiments, we loaded Cloverleaf3D vector field data from disk and set the maximum simulation step size to 0.01 and used 800 simulation steps. Therefore, we generated pathlines for a duration of 800 simulation steps by stitching together results using successive batches of basis flows [5, 13, 7].

Figure 27 groups configurations by number of processes used, i.e., the degree of decomposition. We note that Lagrangian-MPI accuracy is not affected by degree of decomposition and hence remains constant irrespective of number of processors. When considering only a single processor, both methods lose particles that exit the entire domain during the interval of advection. Thus, performance on a single MPI task is identical for both methods. When decomposed across 8 tasks, we see the error of Lagrangian-BTO increases slightly for configurations that use a 1:27 data reduction. Specifically, accuracy reduces from 100% as accurate as Lagrangian-MPI to 96% as accurate when the interval increases from 25 to 100. For domain decomposition across 32 tasks, i.e., the highest degree of decomposition we consider, Lagrangian-BTO error increases as the interval increases and fewer particles are used. These experiments are valuable in demonstrating the limitations of the Lagrangian-BTO method. However, we believe our strong scaling study tests an extreme case, because we expect our target applications will have higher resolution grids and use a larger number of particles to capture the flow field behavior.

6.5.2 Arnold-Beltrami-Childress (ABC). We used a 3D time-dependent variant of the ABC flow analytic vector field [37]. We used one complete period of the flow for a total of 1000 time steps at a grid resolution of 256^3. For the **WF2** ABC data set experiments, we considered three options for interval (25, 50, 100) and three options for data reduction (1:1:, 1:8, 1:27). All tests use 16 nodes, 64 MPI tasks, with 4 MPI tasks using 4 GPUs on each node. Table 5 contains configuration details, percentage of discarded particles, speed-up achieved, and the reconstruction accuracy percentages for the ABC data set.



(a) Lagrangian-MPI (b) Lagrangian-BTO

Figure 28. Qualitative comparison of post hoc FTLE visualizations generated using basis flows for the ABC data set.

Interval	Reduction	L-BTO (s)	L-MPI (s)	Speed-up	Discarded %	Greatest Max L2-norm	Avg Max L2-norm	Total Avg L2-norm	Accuracy %
5	1:1	0.0062	0.0089	**1.4x**	0.7	5.66×10^{-4}	1.73×10^{-4}	1.89×10^{-5}	**99.7**
10	1:1	0.0035	0.0089	**2.5x**	2.7	3.12×10^{-3}	1.20×10^{-3}	1.45×10^{-4}	**98.1**
5	1:8	0.0024	0.0044	**1.8x**	0.8	8.64×10^{-4}	2.69×10^{-4}	3.77×10^{-5}	**99.5**
10	1:8	0.0022	0.0059	**2.6x**	2.1	4.41×10^{-3}	1.45×10^{-3}	2.06×10^{-4}	**97.3**
5	1:27	0.0031	0.0038	**1.2x**	0.7	8.47×10^{-4}	3×10^{-4}	4.32×10^{-5}	**99.4**
10	1:27	0.0024	0.0094	**3.9x**	1.9	4.27×10^{-3}	1.72×10^{-3}	2.48×10^{-4}	**96.8**
5	1:64	0.0025	0.0037	**1.4x**	0.5	9.19×10^{-4}	3.30×10^{-4}	4.34×10^{-5}	**99.4**
10	1:64	0.0029	0.0039	**1.3x**	1.5	4.89×10^{-3}	1.96×10^{-3}	2.80×10^{-4}	**96.4**

Table 6. Speed-up and reconstruction accuracy results for the Jet flow data set **WF2** experiments. Lagrangian-BTO and Lagrangian-MPI columns show average time per step in seconds. Reported results are measured across all intervals.

Interval	Reduction	L-BTO (s)	L-MPI (s)	Speed-up	Discarded %	Greatest Max L2-norm	Avg Max L2-norm	Total Avg L2-norm	Accuracy %
10	1:1	0.0027	0.0067	**2.4x**	4.1	5.06×10^{-4}	1.52×10^{-4}	3.93×10^{-5}	**99.5**
20	1:1	0.0026	0.0067	**2.5x**	8.4	8.19×10^{-4}	4.83×10^{-4}	1.52×10^{-4}	**98.0**
40	1:1	0.0026	0.0139	**5.2x**	15.9	3.97×10^{-3}	2.25×10^{-3}	9.02×10^{-4}	**88.5**
50	1:1	0.0025	0.0079	**3.1x**	19.3	6.04×10^{-3}	3.59×10^{-3}	1.64×10^{-3}	**79.1**
10	1:8	0.0024	0.0045	**1.8x**	3.2	9.68×10^{-4}	1.67×10^{-4}	3.96×10^{-5}	**99.4**
20	1:8	0.0023	0.0107	**4.6x**	7.9	1.74×10^{-3}	7.05×10^{4}	2.19×10^{-4}	**97.2**
40	1:8	0.0027	0.0046	**1.7x**	15.5	6.19×10^{-3}	2.92×10^{-3}	1.18×10^{-3}	**85.6**
50	1:8	0.0030	0.0045	**1.4x**	18.9	7.74×10^{-3}	4.32×10^{-3}	2.02×10^{-3}	**74.3**

Table 7. Speed-up and reconstruction accuracy results for the Nyx simulation data set **WF2** experiments. Lagrangian-BTO and Lagrangian-MPI columns show average time per step in seconds. Reported results are measured across all intervals.

Lagrangian-BTO demonstrates an average of 2.3x speed-up when compared to Lagrangian-MPI. For the ABC data set, reconstruction accuracy does not significantly deteriorate when using a smaller number of particles. However, for configurations with an interval equal to 100, a large number of particles exit the node boundaries and are terminated. For example, when the interval is set to 100 and a data reduction of 1:1 is used, over 1.7M particles are terminated every interval (approximately 10% of all seeds initially placed), which results in the reconstruction accuracy reducing to 95.5%. For the ABC data set, the number of particles terminated per interval approximately doubles every time the interval doubles. We observe that all configurations that have an interval of 25 or 50 achieve a reconstruction accuracy that is greater than 99%. When considering the average L2-norm error of individual intervals of the ABC data set in Figure 31c and 31d, we observe a sinusoidal behavior in the error that is due to the sinusoidal ABC analytical function.

Figures 28a and 28b compare FTLE visualizations generated using approximately 2M basis flows computed over 100 time steps and saved by each technique (configuration in row 6 of Table 5). Although we observe small discontinuities in the ridges of the FTLE field generated using Lagrangian-BTO basis flows, the overall quality of the FTLE visualization is similar.

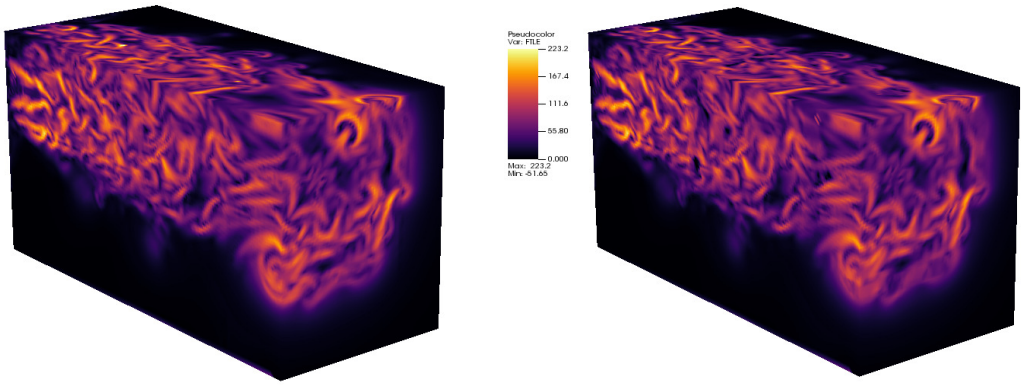
6.5.3 Jet Flow Simulation. The Jet data set is a simulation of a jet of high-velocity fluid entering a medium at rest. It was created using the Gerris Flow Solver [66]. The vector field is defined over a $128 \times 256 \times 128$ uniform grid and we use a total of 500 time steps (previously subsampled). These **WF2** experiments tested the performance of the Lagrangian-BTO when reconstructing this turbulent data set by considering two intervals (5, 10) and four data reduction values (1:1,

1:8, 1:27, 1:64). All tests used 16 nodes, 64 MPI tasks, with 4 MPI tasks using 4 GPUs on each node. Table 6 contains configuration information and results of the Jet data set.

Lagrangian-BTO demonstrates an average speed-up of 2x when compared to Lagrangian-MPI for this data set. The domain contains regions of high velocity resulting in the reconstruction accuracy of the Lagrangian-BTO analysis filter being adversely affected as the interval increases. For configurations with the interval set to 5, the data reduction is less consequential, and all configurations achieve greater than 99% accuracy. However, for configurations with the interval set to 10, the reconstruction accuracy decreases from 98.1% to 96.4% as data reduction ranges from 1:1 to 1:64. Thus, similar to the result observed in the strong scaling study in Section 6.5.1, the combination of larger interval and reduced number of particles results in less accurate reconstruction.

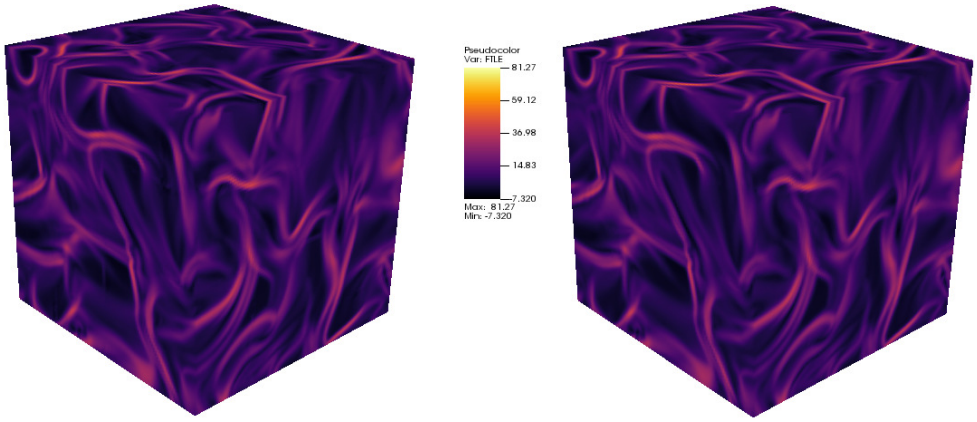
Figures 29a and 29b compare visualizations produced using 4.2M basis flows saved by each method (configuration in row 2 of Table 6). The Lagrangian-BTO FTLE visualization has some instances of a less pronounced structure in the FTLE ridges. However, the differences are localized and relatively small compared to the overall information conveyed by the images.

6.5.4 Nyx Cosmology Simulation. The Nyx simulation is a N-body and gas dynamics code for large-scale cosmological simulations [67]. We use a Nyx test executable named TurbForce to generate 500 cycles of a 128^3 data set with an average time step of 0.002. The generated flow field grows in turbulence over the duration of the simulation. These **WF2** experiments with the Nyx data set test the reconstruction accuracy of Lagrangian-BTO when considering four intervals (10, 20, 40, 50) and two data reduction values (1:1, 1:8). Similar to previous setups, we use



(a) Lagrangian-MPI (b) Lagrangian-BTO

Figure 29. Qualitative comparison of post hoc FTLE visualizations generated using basis flows for the Jet flow data set.



(a) Lagrangian-MPI (b) Lagrangian-BTO

Figure 30. Qualitative comparison of post hoc FTLE visualizations generated using basis flows for the Nyx data set.

64 MPI tasks evenly distributed across 16 nodes with each MPI task using a single GPU. Table 7 details test configurations and results for the Nyx data set.

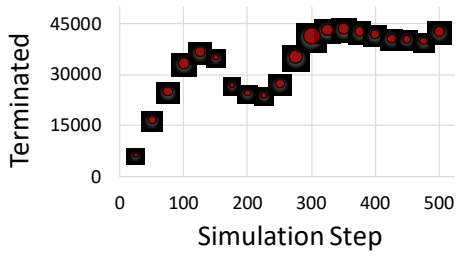
Lagrangian-BTO demonstrates an average speed-up of 2.8x over Lagrangian-MPI. Configurations with an interval of 10 or 20 have high reconstruction accuracies greater than 97%. However, as the interval increases, the reconstruction accuracy is adversely affected by the large number of particles terminated and the significant turbulence in this data set. Further, Lagrangian-BTO can reconstruct

the field relatively more accurately when using a 1:1 data reduction compared to a 1:8 configuration. Figures 30a and 30b compare FTLE visualizations generated using approximately 2M basis flows saved by each technique (configuration in row 1 of Table 7). We observe qualitatively comparable FTLE ridge structures.

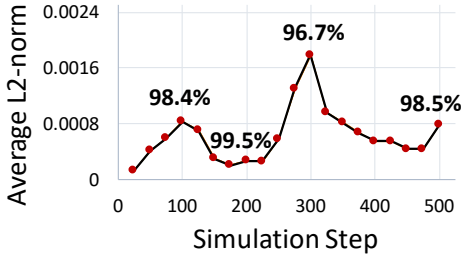
6.6 Conclusion

Our proposed algorithm represents an important step in enabling the **L-ISR-PHE** paradigm to be viable for large-scale simulations. Predecessor work demonstrated that accuracy-storage tradeoffs were clearly superior to the traditional approach [5]. However, this work did not place a significant emphasis on minimizing in situ execution times and ensuring scalability. Our work addresses this point, and the corresponding reduced execution times (2x to 4x) and improved scalability remove another barrier to adoption.

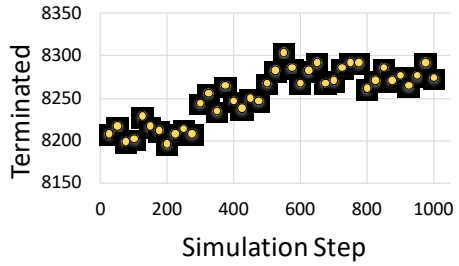
We feel the most surprising result from our work is the rate at which we achieve high reconstruction accuracy. Clearly, terminating particles at block boundaries makes for less useful basis flows, and can create issues where post hoc exploration accuracy suffers. However, our results empirically show that this feared case happens relatively infrequently in practice and is limited to instances of long intervals, i.e., large integration times. Our proposed approach works particularly well for practical configurations with a short or medium interval, while delivering the same performance benefits. 31 of our 35 tests gave accuracy that was greater than 96%. In addition to our quantitative evaluation, we qualitatively showed that comparable FTLE visualizations can be generated using basis flows extracted at a fraction of the in situ cost. These evaluations are relative to the Agranovsky et al. approach, which incurred significantly more cost in execution time and scales poorly. Further, while we are slightly less accurate than Agranovsky et al., our



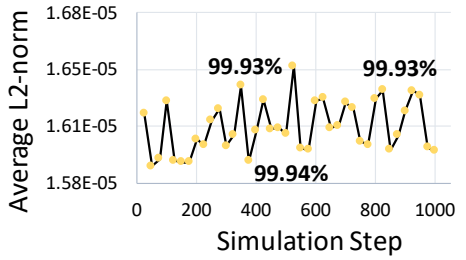
(a) Cloverleaf3D



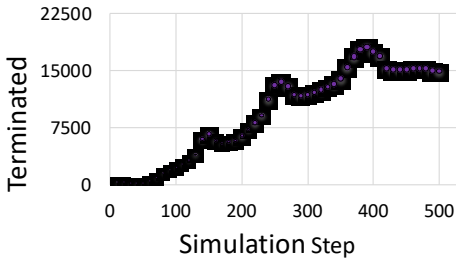
(b) Cloverleaf3D



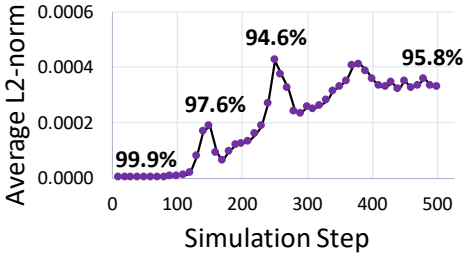
(c) ABC



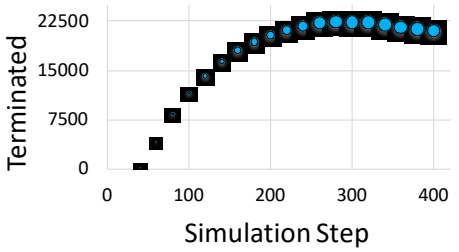
(d) ABC



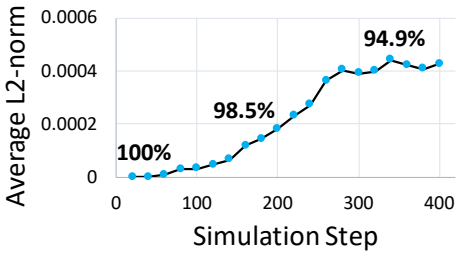
(e) Jet Flow



(f) Jet flow



(g) Nyx



(h) Nyx

Figure 31. The plots show the relation between number of particles terminated and average L2-norm for all intervals of a single configuration of each data set. 31a and 31b show row 10 of Table 4 for the Cloverleaf3D data set. 31c and 31d show row 7 of Table 5 for the ABC data set. 31e and 31f show row 4 of Table 6 for the Jet flow data set. 31g and 31h show row 6 of Table 7 for the Nyx data set.

approach would still be much more accurate than the traditional Eulerian approach for time-varying flow visualization in sparse temporal settings.

In terms of future work, we aim to store particle trajectory termination locations on the boundary, develop Lagrangian-based advection schemes that can consume flow maps with trajectories stopping (and starting) at arbitrary times, integrate adaptive variable duration variable placement (VDVP) techniques [8], and consider the use of flow field characteristics to guide flow map extraction. Further, research is required to address edges cases and the limitations of a communication-free model demonstrated by our experiments. We believe our work is foundational because it evaluated the use of a simple communication-free model and demonstrated improved scalability with only a reasonable loss of accuracy. Remaining communication-free is particularly essential for Lagrangian analysis techniques that aim to scale well and remain within in situ constraints of large-scale simulations.

CHAPTER VII

IN SITU VECTOR FIELD DATA REDUCTION VIA LAGRANGIAN REPRESENTATIONS ON SUPERCOMPUTERS

Most of the text in this chapter comes from a manuscript in submission, which is a collaboration between Hank Childs and myself. I was responsible for system implementation, conducting the experiments, and preparing the manuscript. Hank Childs contributed text to the manuscript, in addition to advising and providing extensive feedback throughout.

7.1 Introduction

This chapter presents an empirical study on the **L-ISR-PHE** workflow to the **EUS** problem. It considers **L-ISR-PHE** holistically: considering evaluation criteria at each phase of processing, defining metrics to measure these criteria, defining workloads of interest, and then evaluating the workloads and metrics in a large study. One highlight of our study is that it is the first to evaluate the encumbrance placed on a simulation code during *in situ* reduction. Of note, all previous studies ran in "theoretical" in situ environments, meaning that they loaded data sets from disk, rather than truly integrated with a simulation code. Another highlight is that our study provides detailed quantitative evaluations of the extracted Lagrangian data, as well as an estimate of costs for distributed memory *post hoc* reconstruction. Previous studies have failed to consider the distribution of outcomes when inferring new particle trajectories, instead focusing on average behavior. In all, our empirical study provides the most clear evidence to date that **L-ISR-PHE** is viable and should be the preferred solution for most **EUS** problems.

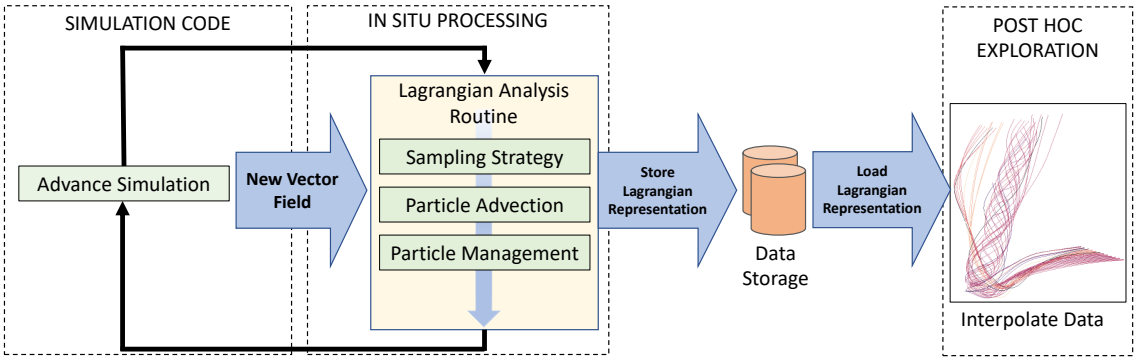


Figure 32. Schematic diagram of the **L-ISR-PHE** workflow showing *in situ* processing, data storage, and *post hoc* exploration.

7.2 Related Work

Time-dependent vector field exploration poses several large data analysis and visualization problems. Often, solutions to large data challenges involve data reduction. Multiple works have proposed vector field reduction strategies while maintaining an Eulerian representation. Lodha et al. [68] controlled the compression of similar vectors into single vectors representing larger area. Further, Lodha et al. [69] proposed a top-down topology preserving compression technique. Theisel et al. [70] computed critical points and viewed the task as a mesh reduction, and later provided a threshold to filter important features [71]. Each of these studies performed compression on the vector field of a single time step. With the objective of highlighting temporal features of the vector field, Tong et al. [72] compressed the total amount of data steps stored by identifying key time steps. Although these techniques could be used to reduce data and store more frequently, these approaches do not address the challenge of increasing temporal sparsity.

7.3 Evaluating L-ISR-PHE

This section describes considerations for an empirical study of the **L-ISR-PHE** workflow. Specifically:

- Subsection 7.3.1 describes the instantiation we consider.
- Subsection 7.3.2 describes evaluation criteria.
- Subsection 7.3.3 describes important factors in defining workloads.

7.3.1 Instantiation. Figure 32 shows a high-level description of the **L-ISR-PHE** workflow. There are many possible strategies for accomplishing the components within this workflow, i.e., sampling strategy, particle management, storage, and interpolation. That said, we focus this empirical study on the current best practices in this space.

***In Situ* Reduction:** For this empirical study, we introduced particles using the uniform seed placement scheme from Agranovksy et al. [5], including re-introducing particles at fixed intervals. Our particle termination follows the communication-free model from Chapter VI, where particles are terminated either if they reach the end of the interval of if they exit the block. While this captures less information at the block boundaries, results show that the overall loss of accuracy is very low, and the *in situ* encumbrance is much reduced.

Data Storage: For our empirical study, a particle trajectory in the Lagrangian representation is stored using a start and end location of the trajectory during the interval of calculation, similar to previous works [5, 7, 9].

***Post Hoc* Exploration:** The essential operations involved in constructing new particle trajectories are identifying which basis flows to interpolate and performing interpolation. Further, distributed-memory settings require communication to continue particle trajectory computation across node boundaries.

Depending on whether the Lagrangian representation is structured or unstructured, different search structures are required. We believe evaluating the cost of interpolating an unstructured particle set is more valuable, as it informs the general case. Therefore, for our empirical study, we use the Lagrangian-based advection scheme described in Chapter VI, to perform search structure (Delaunay triangulation) construction in parallel and operate in a distributed-memory environment. To perform interpolation on unstructured data, multiple scattered point interpolation schemes can be used. However, the choice of interpolation scheme impacts efficacy, and thus, we use Barycentric coordinates, as recommended in a study by Agranovsky et al. [19].

7.3.2 Evaluation Criteria. We consider evaluation criteria for each component of the workflow:

***In situ* reduction:**

- **ISR-1 Time:** the execution time spent by the simulation on data analysis and visualization.
- **ISR-2 Memory:** the runtime memory used by *in situ* processing.

Data storage:

- **DS-1 Size:** the file storage costs (i.e., bytes).

***Post hoc* exploration:**

- **PHE-1 Time:** the execution time spent to construct new particle trajectories from basis flows (search structure construction, interpolation, communication).
- **PHE-2 Accuracy:** the accuracy of interpolated trajectories.

We also eliminated some criteria from consideration to limit scope:

***In situ* reduction:** We did not add an evaluation criteria to consider the ease of *in situ* integration. This is an important concern, but we feel that it is beyond the scope of this study. Further, there has been significant research on reducing the burden on simulation codes to incorporate *in situ* visualization routines [73, 74, 75, 76, 77], and so we are confident that this barrier will become smaller over time.

Data storage: We did consider and measure execution time, but found supercomputing I/O times were very fast for our scale of study and often contained noise, likely due to contention on the supercomputer. More discussion can be found in 7.4.5.2.

***Post hoc* exploration:** We did not consider if and whether Lagrangian-based extracts would affect specific flow visualization techniques. For example, are techniques, such as path surfaces or finite-time Lyapunov exponents, sensitive or insensitive to the **L-ISR-PHE** workflow? We view this as future work.

7.3.3 Workload Factors. To understand the technique performance characteristics of the **L-ISR-PHE** workflow, we identified four parameters that when varied produce the workloads we want to evaluate for our empirical study.

– **Number of particles:** Our study varies the number of particles initialized per node and thus inform the cost of performing particle advection for varying workloads every cycle of the simulation. Further, the number of particles initialized is directly impacts the size of the data stored to disk and the accuracy of the reconstruction. We specify the number of particles initialized using the notation **1:X**, where X is the reduction factor. For example, a 1:1 configuration states that one particle is used for every grid point (no

reduction) and a 1:8 configuration states that one particle is used for every 8 grid points (12.5% of the original data size).

Impacts → ISR-1, **ISR-2**, **DS-1**, **PHE-1**, **PHE-2**

– **Interval:** We consider the interval or frequency at which files are stored to disk. For a given total number of simulation cycles, this impacts the total amount of data stored to disk. Additionally, for the Lagrangian representation, the interval is equal to the integration length of each particle, and can thus, be consequential to the accuracy of reconstruction.

Impacts → DS-1, **PHE-1**, **PHE-2**

– **Grid size:** We consider different grid sizes to measure the *in situ* encumbrance of varying workloads. Different grid sizes will use a different number of particles to sample the domain reasonably accurately. In particular, we are interested in the *in situ* encumbrance when a single compute node is operating on a large number of grid points. An additional benefit of varying grid size is insight into the variation in simulation cycle time and consequently the percentage of time spent on *in situ* processing.

Impacts → ISR-1, **ISR-2**, **DS-1**

– **Concurrency:** We consider the costs at various scale (i.e., number of compute nodes, MPI ranks). Further, the simulation codes required different parallelization hardware and thus, across simulation codes we measure the costs of Lagrangian representation extraction using, both, GPUs and CPUs for particle advection.

Impacts → ISR-1, **PHE-2**

7.4 Empirical Study Overview

This section provides an overview of our experiments. It is organized as follows: experiments performed (7.4.1), simulation codes (7.4.2), software implementation (7.4.3), hardware (7.4.4), and metrics (7.4.5).

7.4.1 Experiment Overview. Our experiments are designed in response to the evaluation criteria from Section 7.3.2. Since our evaluations can be separated into two distinct phases, we organized our experiments into two distinct campaigns (one for *in situ* encumbrance and one *post hoc* efficacy), although some of the experiments were used in both campaigns.

Our experiments considered five basic factors:

- Simulation code
- Number of particles (Lagrangian basis flows)
- Interval (number of cycles between saves)
- Grid size
- Concurrency

For each factor, there are many possible options. Therefore, running experiments for the cross-product of options was prohibitive, especially since we had limited time on our supercomputer (1000 node hours). Instead, we sampled the space of possible options. For both campaigns, our organization was around our three simulation codes: Cloverleaf3D, SW4, and Nyx (described in subsection 7.4.2). For a given simulation code, we varied some factors and fixed others. Our goal was to simultaneously provide coverage and yet allow us to see the impact of certain factors, all while staying within our compute budget. In all, we ran 47 experiments, 22 for *in situ* encumbrance and 25 for *post hoc* efficacy. Table 8 shows our choices for the *in situ* encumbrance campaign, while Table 9 shows our choices for the *post*

hoc efficacy campaign. Specific choices for options are documented in the Results section.

Simulation Code	Cloverleaf3D	SW4	Nyx
# of Particles	3	3	3
Interval	3	1	1
Grid Size	1	3,2	2
Concurrency	1	2	1
Total Experiments	9	7	6

Table 8. Experimental overview for the *in situ* encumbrance campaign. For SW4, we were able to run a very fine grid size at low concurrency, but not the entire cross product of options due to limitations in compute time. Overall, we considered 22 experiments for this campaign.

Simulation Code	Cloverleaf3D	SW4	Nyx
# of Particles	3	4	3
Interval	3	1	4
Grid Size	1	1	1
Concurrency	1	1	1
Total Experiments	9	4	12

Table 9. Experimental overview for the post hoc efficacy campaign. Overall, we considered 25 experiments for this campaign.

7.4.2 Simulation Codes. For our study we consider three simulation application codes that are used and/or developed as part of the Exascale Computing Project from the United States Department of Energy.

First, we use the Cloverleaf3D [65] mini or proxy ECP application that solves compressible Euler equations in a hydrodynamics setting on a Cartesian grid using an explicit second-order method. Cloverleaf3D has been developed and used by several studies to evaluate emerging architectures and various techniques targeting Exascale applications. The simulation is initially relatively stable and begins with an energy bar expanding from the center of the XY plane along the Z-axis. Figure 34 show pathlines calculated in the Cloverleaf3D domain that show this initial behavior in the simulation.

Next, we consider the SW4 seisomology simulation [18]. This is an ECP application developed to study seismic wave propagation. It operates and produces multiple domains with a time-dependent displacement field depending on the input deck provided to it. We operate on a single domain and use the displacement vector field as input to our *in situ* Lagrangian operator. Figure 33 is generated by visualizing the displacement magnitude of the particle trajectories extracted over the first 1000 cycles.

The last data set we consider is the Nyx cosmology simulation [67], another ECP application. The simulation's hydrodynamics is based on a compressible flow formulation in Eulerian coordinates. We built an Lya executable used to model Lyman-alpha forest in quasar spectra. For this simulation, we derived the velocity field using the fields of momentum and density. Figure 35 is a visualization of the volume of the domain using 50,000 randomly seeded pathlines integrated for the first 25 cycles of the simulation.

7.4.3 Software Implementation.

7.4.3.1 In Situ Reduction. We use the Ascent [59] *in situ* infrastructure and VTK-m [58] library to implement *in situ* data reduction via by calculating a Lagrangian representation. The VTK-m Lagrangian filter on each rank operates independently and maintains its own list of basis particles and uses the existing particle advection infrastructure available in VTK-m [60]. RK4 particle advection is implemented using VTK-m worklets (kernels or functors) that offer performance portability by utilizing the underlying hardware accelerators. In our implementation, each Lagrangian filter stores the displacement of each particle (3 doubles), as well as its validity (1 Boolean), i.e., whether the particle remained within the domain during the interval of calculation. In more complicated

frameworks, it is possible to associate additional information (for example, ID, age, start location, previous locations, etc.) with each particle at the cost of higher runtime memory usage and data storage.

We use Ascent to store the complete velocity field at a specified frequency in order to evaluate the traditional Eulerian paradigm. For every Eulerian configuration, we store the full spatial resolution of the simulation domain under consideration.

7.4.3.2 Post Hoc Exploration. We build two parallelize-over-data distributed-memory *post hoc* interpolation pipelines, one for each: Lagrangian and Eulerian. For the Lagrangian **PHE** we construct a search structure in the form of a Delaunay triagulation over the start locations of valid basis particles using CGAL [61] to identify particle neighborhoods. In our implementation, each rank loads basis particles of the rank itself as well as basis particles generated by spatially adjacent ranks (i.e., upto 27 ranks for a rectilinear simulation grid). Once a particle neighborhood is identified, barycentric coordinate interpolation is used to calculate the displacement of p. Our implementation uses MPI to communicate particles across ranks and continue the integration of trajectories across node boundaries.

For the Eulerian *post hoc* interpolation pipeline, we use the VTK-m particle advection infrastructure to perform RK4 integration and MPI for communication of particles across ranks.

7.4.4 Runtime Environment. Our empirical study uses the Summit supercomputer at ORNL. A Summit compute node has two IBM Power9 CPUs, each with 21 cores running at 3.8 GHz and 512 GBytes of DDR4 memory. Nodes on Summit also have enhanced on-chip acceleration with each CPU

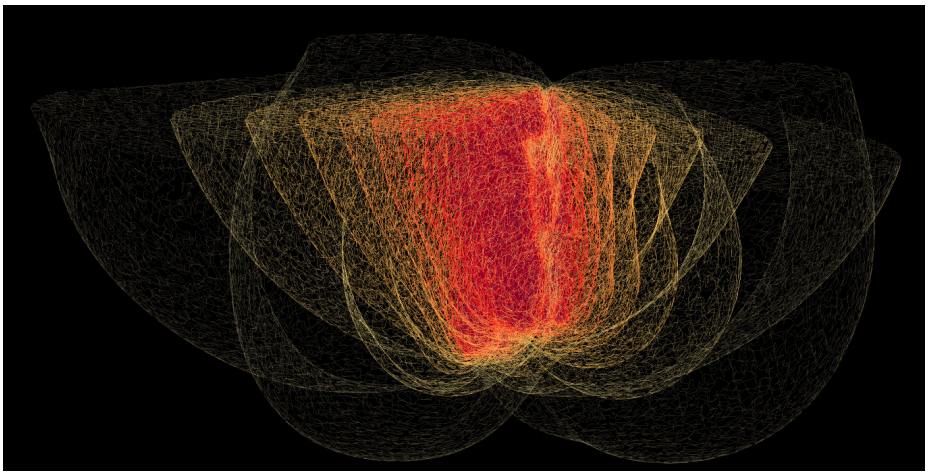


Figure 33. SW4 wave propagation visualization.

connected via NVLink to 3 GPUs, for a total of 6 GPUs per node. Each GPU is an NVIDIA Tesla V100 with 5120 CUDA cores, 6.1 TeraFLOPS of double precision performance, and 16 GBytes of HBM2 memory. Lastly, it uses a Mellanox EDR 100G InfiniBand, Non-blocking Fat Tree as its interconnect topology.

7.4.5 Evaluation Metrics.

7.4.5.1 In Situ Encumbrance. Our empirical study measures *in situ* encumbrance in terms of execution time and memory usage. For **ISR-1**, we measure the cost of each invocation of the Lagrangian VTK-m filter and report the average time, i.e., cost of particle advection for one cycle or **Step**. Additionally, we measure and report the percentage of simulation time spent on data analysis and visualization routines, or **DAV%**. For this we consider the simulation cycle time or Sim_{cycle}. For **ISR-2**, we measure the runtime memory cost incurred by every compute node to maintain the state (current position) of particles at runtime in Bytes.

7.4.5.2 I/O Cost. In this empirical study, we do not report or factor the cost of I/O write times. Besides being an operation that is performed

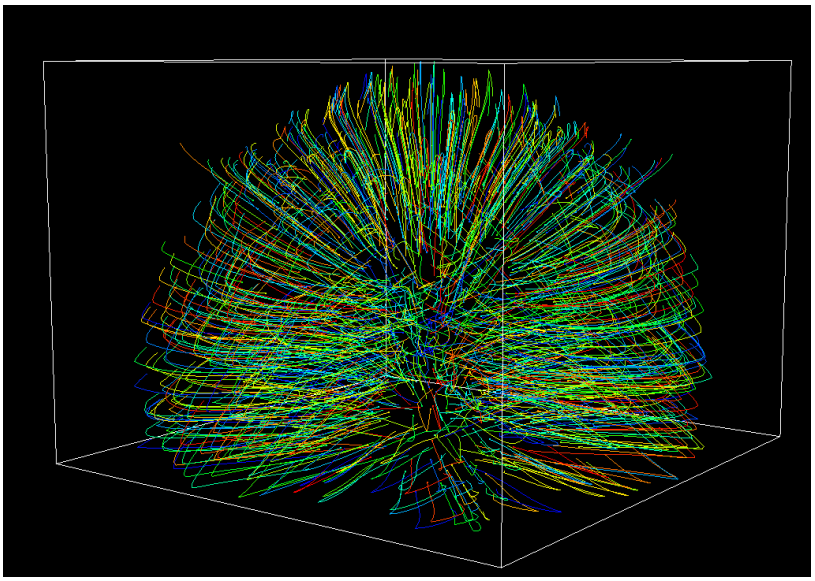


Figure 34. Visualization of pathlines in the Cloverleaf3D domain.

infrequently in our case, we observed for the scale of study we conducted that Summit provides very fast write times. Table 10 documents write times of varying file sizes in binary format on Summit. Given the range of file sizes stored to disk by

File Size (MB)	1 MPI/Node (s)	6 MPI/Node (s)
1	0.0018	0.0022
10	0.0032	0.0045
50	0.0064	0.013
100	0.0125	0.038
200	0.0231	0.171

Table 10. Write time measurements for various file sizes (in binary format) to disk on Summit. We consider two cases: one MPI rank or six simultaneous MPI ranks each writing the file to disk on a single compute node. For each timing, we average over multiple runs and every timing is reported in seconds.

a single MPI rank in our empirical study is between 0.5 MB to 115 MB, we believe this cost is negligible at the scale that we test (and perhaps, even at larger scales). Thus, we limit our measurement and discussion of I/O to the total data storage required on the file system and report **DS-1** in Bytes stored.

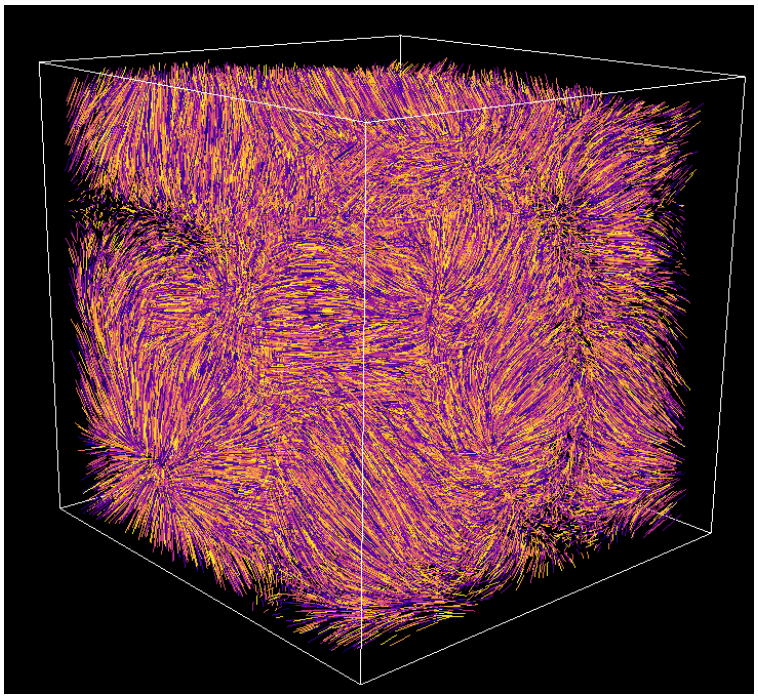


Figure 35. Nyx flow visualization: 50,000 randomly seeded pathlines calculated for the first 25 cycles of the simulation.

7.4.5.3 Post Hoc Efficacy. Our empirical study measures time-dependent vector field reconstruction error by evaluating the accuracy of test particles trajectories interpolated using the extracted Lagrangian representation.

For **PHE-1**, our empirical study measures the L2-norm, i.e., the Euclidean distance, for each interpolated point and compares it to a ground truth that is precomputed using the complete simulation data. We use $\mathbf{Avg}_{L2}$ and $\mathbf{Max}_{L2}$ to denote the average and maximum L2-norm for an individual particle trajectory, respectively.

An overall average of $\mathbf{Avg}_{L2}$, denoted by $\mathbf{AvgN}_{L2}$, is measured across N test particle samples and provides a robust statistic [5, 7, 8, 78]. Unlike $\mathbf{AvgN}_{L2}$, a maximum error is more susceptible to outliers that could arise from small but complex regions of flow. To provide a more detailed quantitative analysis compared

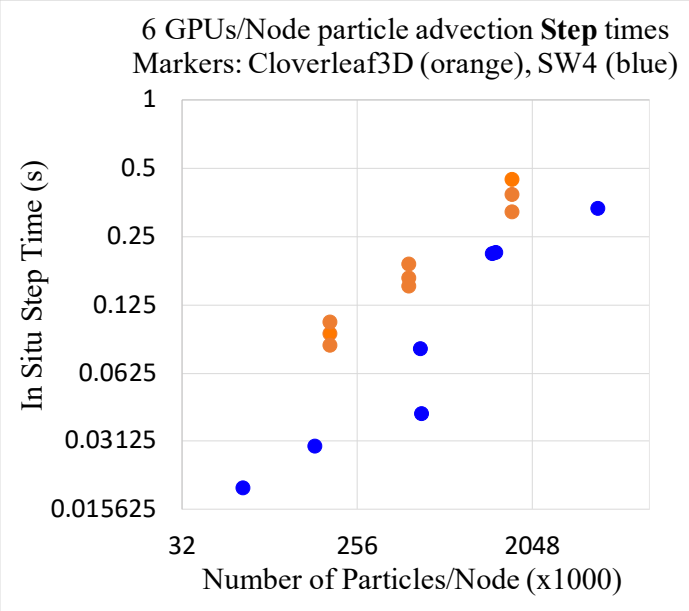
to prior work, we use histograms to capture the distribution of error ($\mathbf{Avg}_{L2}$, $\mathbf{Max}_{L2}$) across all test particles and provides insight into per particle outcomes. Further, for **PHE-2** we include a study of *post hoc* reconstruction costs.

7.5 Results and Discussion

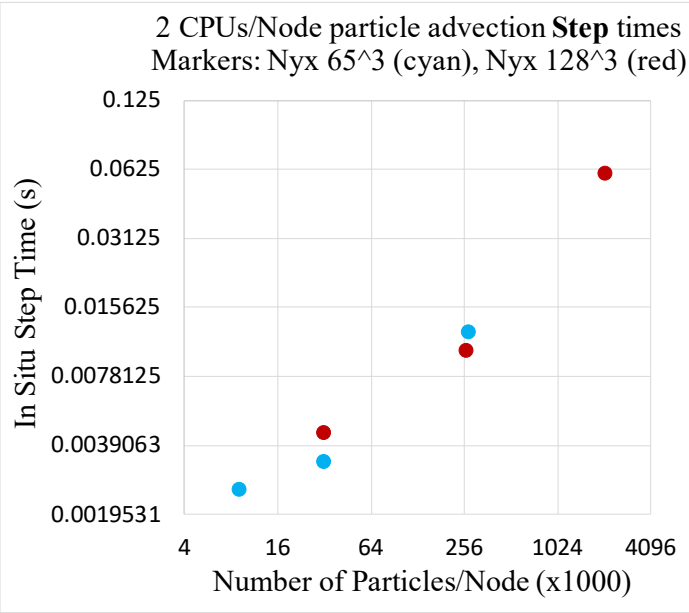
We evaluated **L-ISR-PHE** using seismic wave propagation, cosmology, and proxy ECP hydrodynamics simulations (Section 7.4.2) for two aspects: (1) *in situ* encumbrance under varying workloads (Section 7.5.1); and (2) *post hoc* efficacy for various configurations (Section 7.5.2).

7.5.1 In Situ Encumbrance. Table 11 contains the results of our experiments for this campaign using all three simulation codes. In this discussion, we assume a simulation can afford to spend 10% to 20% on *in situ* processing routines and refer to this as the **budget**. Although this might not hold true for every simulation, this estimate is based on interactions with computational scientists and thus, we believe this is a reasonable working estimate. The **Step** and **DAV%** columns in Table 11 redundantly encode the value in each cell using cell background color (white to pure red hue for the ranges [0,0.75] (**Step** in seconds) and [0,20] (**DAV%**), respectively).

For **ISR-2**, i.e., memory costs, we observe that across all experiments, the largest usage of runtime memory was approximately 112 MB. Each Summit node has multiple GBs of memory on CPU (512) and GPU (16), and we believe extracting a Lagrangian representation increases the cost of memory on the simulation by approximately one simulation "field." We note that simulations can have tens to hundreds of fields defined on the simulation grid and thus, this cost would likely be considered acceptable for most simulations. Our reporting of memory usage is contained in Table 11.



(a) GPU timings



(b) CPU timings

Figure 36. GPU and CPU timings - Number of particles vs time required per step.

Nodes	MPI Ranks	Dims	Interval	Sim_{cycle}	Particles /Node	Memory /Node (MB)	Step	DAV%
Cloverleaf3D Proxy Hydrodynamics Application								
16	96	586^3	20	4.73	1.5M	40.2	0.4475	9.408
			40	4.08			0.3221	7.894
			60	4.39			0.3838	8.742
			20	4.50	474k	12	0.1882	4.182
			40	4.14			0.1628	3.932
			60	4.33			0.1498	3.459
			20	4.19	186k	4.2	0.0925	2.207
			40	4.11			0.1043	2.537
			60	3.87			0.0830	2.144
SW4 Seismic Modeling Simulation								
1	6	$251^2 \times 70$	200	0.35	555k	13.89	0.0412	11.67
		$335^2 \times 93$		2.02	1.3M	33.16	0.2125	10.48
		$501^2 \times 139$		7.58	4.4M	111.13	0.3309	4.365
64	384	$1001^2 \times 276$		1.6	66k	1.6	0.0194	1.201
				1.5	146k	3.6	0.0295	1.944
				1.3	540k	13.5	0.0798	6.175
		$1335^2 \times 368$		2.9	1.2M	31.9	0.2095	7.074
Nyx Cosmology Simulation								
1	1	65^3	100	10.9	274k	6.8	0.0122	0.112
					32k	0.8	0.0033	0.030
					9k	0.2	0.0025	0.023
		129^3		88.3	2.1M	53.6	0.0596	0.067
					262k	6.5	0.0101	0.011
					32k	0.8	0.0044	0.005

Table 11. *In situ* encumbrance evaluation and experiment configurations for our three simulation codes.

7.5.1.1 Cloverleaf3D Hydrodynamics Proxy Simulation.

For the Cloverleaf3D simulation, we considered 3 options for number of particles and interval, and 1 option for grid size and concurrency. In particular, we are interested in the *in situ* encumbrance (**ISR-1**) of varying particle advection workloads, i.e., the number of particles. We note that each node used 6 GPUs for particle advection and that they all access the same shared memory. For our specific grid

size and domain decomposition, each MPI rank operated on over 2M grid points and the Sim_{cycle} was usually between 4-5 seconds. Overall, we observe an increase in **Step** costs as the number of particles advected per node increases. Given, the Sim_{cycle} remained relatively stable, the increase in **Step** is clearly matched by the **DAV%** trend.

For this integration, we observed that as the number of particles increases from 186k to 474k (2.5X increase) that the cost of performing particle advection only increases by approximately 1.6X. For the next workload increase, i.e., sampling 1.5M particles (∼3X increase), the cost of performing particle advection increases by approximately 2X. By running the same workload multiple times, we capture variation in the costs within a workload. The variation in the **Step** cost is greater when the workload is larger and we attribute this to the increased memory allocation and memory transfer costs each step. This is relevant particularly on GPUs where the initial setup cost can be high.

Overall, for **ISR-1**, we found that for our set of experiments increasing the number of particles by 8X results in the *in situ* encumbrance increasing by 3X-4X with the Sim_{cycle} relatively stable. The cost of a single **Step** to calculate the Lagrangian representation for Cloverleaf3D was as low as 0.08 seconds and in all cases, below half a second, thus, remaining within our identified **budget**.

7.5.1.2 SW4 Seismic Wave Propagation Simulation. For the SW4 simulation, we considered 2 concurrencies: 1 compute node (6 MPI ranks, GPUs) and 64 compute nodes (384 MPI ranks, GPUs).

In the first case, i.e., using 1 compute node and 6 MPI ranks, we considered three grid sizes, each using a proportional number of particles (1:8). We increase the number of particles proportionately, rather than holding it constant, since we

believe this would be a more representative of a workload. These results in our empirical study highlight the impact of an increasing grid size on **ISR-1** and the relation to Sim_{cycle}. For the smallest grid size, 555k particles per node are advected every cycle. Although the cost of a particle advection **Step** is low (0.041s), the **DAV%** is over 10% because the Sim_{cycle} is very small (0.035s) in this case. In contrast, for the largest grid size (each rank operated on 5.8M grid points), we advected 4.4M particles per node and observed a proportional increase in **Step** cost, but half as much time was spent by the simulation on **DAV%**. This is due to the higher Sim_{cycle} for the larger grid size. We note this trend would be expected for computational simulations as they increase in resolution per compute node.

In the second case, i.e., using 64 compute nodes and 384 MPI ranks, we ran SW4 four times. Three times with one grid size to observe *in situ* encumbrance for varying particle advection workloads, and one time using a larger grid with 1:8 particles per node. Similar to the Cloverleaf3D experiments, we observed a steady increase in **Step** and **DAV%** as the number of particles per node increases. For the fixed grid size, an 8X increase in the particle advection workload results in an approximately 4X increase, considering Sim_{cycle} with small variability. However, as we increased the grid size, and consequentially, the workload from 540k to 1.2M particles per node ($\sim$2X), although the **Step** cost increased by over 2.6X, **DAV%** increased by less than 1%.

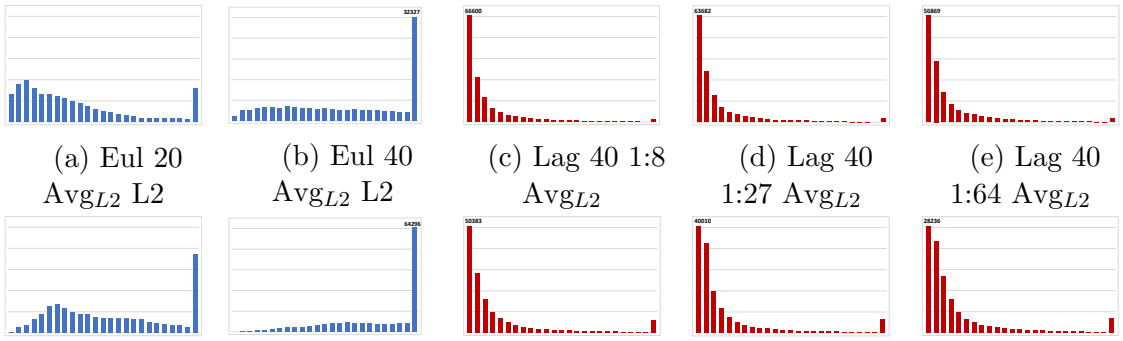
Overall, we first observed that the **DAV%** is closely related to the Sim_{cycle}. Although extracting a Lagrangian representation might place a higher encumbrance on a simulation with a small Sim_{cycle} value, for all grid sizes considered the *in situ* encumbrance, i.e., **DAV%**, of the corresponding workload remained within our expected **budget** and the cost of **Step** was less than half a second in each case.

7.5.1.3 Nyx Cosmology Simulation.

Unlike our previous experiments, the Nyx simulation and Lagrangian filter use OpenMP for parallelism, i.e., particle advection is performed using all the CPU cores on a compute node. We considered 3 options for number of particles and 2 options for grid size.

First, focusing on the impact of an increase in the grid size on **ISR-1**, we found a small increase (<1.5X) in the absolute cost of a particle advection **Step** for the same workload, albeit interpolating a grid 8X in size. Further, in the context of **DAV%**, the Sim_{cycle} cost increases proportionately to the increase in grid size (8X). Thus, the **DAV%** reduces as the simulation grid size increases. Next, for **ISR-1** across workloads using a fixed grid size, for the smaller grid we observed less than a 5X increase when going from 9k particles to 274k particles per node (30X increase in workload). For the larger grid, a 65X increase in workload resulted in a 13X increase in **Step** time.

The most interesting finding of these experiments was that using the CPUs, a single particle advection step for the number of particles we considered, costs less than 6 GPUs. For example, the **Step** cost for 2.1M particles on 2 CPUs is less than half compared to the **Step** cost for 1.3M and 1.5M particles using 6 GPUs. We do note there are differences, such as 6 GPUs (i.e., 6 MPI ranks) accessing the same memory versus 1 MPI rank on 2 CPUs accessing memory. Although this outcome is likely not surprising (given our knowledge of memory allocation and transfer times for GPUs versus CPUs), this finding certainly encourages future research on how to utilize compute resources if the *in situ* routine frequency is very high (every cycle in our study).



(a) Eul 20 Avg_{L2} L2 (b) Eul 40 Avg_{L2} L2 (c) Lag 40 1:8 Avg_{L2} (d) Lag 40 1:27 Avg_{L2} (e) Lag 40 1:64 Avg_{L2}

(f) Eul 20 Max_{L2} (g) Eul 40 Max_{L2} (h) Lag 40 1:8 Max_{L2} (i) Lag 40 1:27 Max_{L2} (j) Lag 40 1:64 Max_{L2}

Figure 37. **Cloverleaf3D** experiment histograms for 100,000 test particle interpolation errors. Each plot has 25 bins, ranging from 0 to >0.05, with bar height encoding number of particles. Horizontal grid lines mark increments of 5,000.

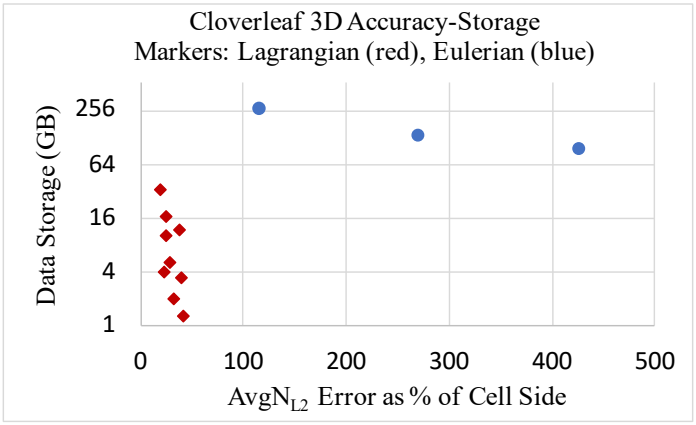


Figure 38. Cloverleaf3D accuracy-data storage scatter plot.

Overall, considering the larger Sim_{cycle} times and low memory latency when parallelizing using CPUs, the highest *in situ* encumbrance we observed to extract a Lagrangian representation was 0.1% of the simulation time.

7.5.2 Post Hoc Efficacy. Table 12 contains the results of our experiments for this campaign using all three simulation codes. For each simulation code we consider multiple options of number of particles and interval. Varying either of these parameters impacts **PHE-1**, **PHE-2**, and **DS-1**. The **Cell**

Technique	Interval	Reduction	Data	AvgN$_{L2}$	Cell Side%
Cloverleaf3D Proxy Hydrodynamics Application					
Eulerian	20	Full Res	267 GB	0.0197	116.17
	40		133 GB	0.0459	270.49
	60		95 GB	0.0725	426.96
Lagrangian	20	1:8	34 GB	0.0032	18.928
		1:27	10 GB	0.0040	23.891
		1:64	4 GB	0.0040	23.583
	40	1:8	17 GB	0.0043	25.646
		1:27	5.1 GB	0.0049	29.145
		1:64	2 GB	0.0053	31.353
	60	1:8	12 GB	0.0064	37.882
		1:27	3.4 GB	0.0066	39.002
		1:64	1.3 GB	0.0070	41.247
SW4 Seismic Wave Modeling Simulation					
Eulerian	250	Full Res	1100 MB	3.5714	0.9224
	500		550 MB	5.0493	1.3023
Lagrangian	250	1:1	1300 MB	0.0005	0.0001
		1:8	158 MB	0.0033	0.0008
		1:27	42 MB	0.0072	0.0018
		1:64	16 MB	0.0128	0.0031
Nyx Cosmology Simulation					
Eulerian	25	Full Res	227 MB	0.010	2.2954
	50		120 MB	0.037	8.4090
	100		67 MB	0.090	20.454
	200		40 MB	0.265	60.227
Lagrangian	25	1:1	232 MB	0.051	11.613
		1:8	27 MB	0.164	37.272
		1:27	8 MB	0.320	72.727
	50	1:1	166 MB	0.059	13.409
		1:8	14 MB	0.153	34.772
		1:27	4 MB	0.256	58.181
	100	1:1	58 MB	0.067	15.227
		1:8	7 MB	0.159	36.136
		1:27	2 MB	0.261	59.318
	200	1:1	29 MB	0.103	23.409
		1:8	3.4 MB	0.204	46.363
		1:27	1 MB	0.321	72.954

Table 12. *Post hoc* efficacy evaluation and experiment configurations for our three simulation codes.

Side% column in Table 12 redundantly encodes the value in each cell using cell background color (white to pure red hue for the range [0,100], where 0 or white indicates a particle is perfectly accurate and 100+ or pure red indicates particles on average are at least a grid cell side away from ground truth). In addition to Table 12, our empirical study includes a more detailed look at per particle interpolation error using histograms. We present a set of histograms for each simulation code. For each histogram chart we exclude the axes and instead describe the common details of the plot in the captions (number of bins, range, horizontal grid line increment, etc.) and use annotation to mark histogram bars whose height exceeds the plot area. Although use of an annotation rather than the true height of the bar visually misrepresents a single data point in some plots, we believe this tradeoff is worth the closer look at the remaining data points.

7.5.2.1 Cloverleaf3D Hydrodynamics Proxy Simulation.

For the Cloverleaf3D time-dependent vector field, we considered 3 options for both number of particles and interval, to encode the behavior of the field. We randomly placed 100,000 test particles in the domain and tested the accuracy of reconstructed trajectories. We use the first 600 cycles of the simulation and set step size to 0.0045. Overall, we observed that the Lagrangian technique performed significantly better and offered improved data storage-accuracy propositions.

With respect to **DS-1** and **PHE-1**, even a 100X data reduction results in improved accuracy compared to storing a full resolution Eulerian grid more frequently. For example, a Lagrangian configuration using 1:64 number of particles and an interval of 60 stores 1.3 GB over 600 cycles, and has an $\mathbf{AvgN}_{L2}$ of 41.2% of the cell side. In comparison, an Eulerian configuration storing the full mesh every 40 cycles requires 133 GB over 600 cycles, and has an $\mathbf{AvgN}_{L2}$ of 270.4%

of the cell side. For all the Lagrangian configurations, the $\mathbf{AvgN}_{L2}$ was low and particles on average remained within the same cell as the ground truth.

Although this proxy simulation demonstrates very clearly the shortcomings of the Eulerian technique as the interval increases, we observed that the Lagrangian technique benefits minimally from an increase in the number of particles. We believe this is due to Cloverleaf3D being a miniapp, where increasing the spatial resolution does not increase the complexity of the physics, i.e., no new features are introduced as they would be in a real-world simulation. That being said, even if the Eulerian technique used multi-resolution to achieve reduced storage, it would be less accurate than Lagrangian, given using the full spatial resolution is less accurate.

The histogram plots in Figure 37 show the distribution of particle interpolation error clearly indicating the superiority of the Lagrangian technique for **EUS**. Comparing the histogram plots, although Eulerian (267 GB) is storing full resolution data sets twice as often, the number of test particles with a $\mathbf{Max}_{L2}$ of over 300% of the cell side distance (right-end bin in each plot) is over 15%, compared to less than 5% for Lagrangian (2 GB to 17 GB) in all cases. This provides intuition regarding the "rate of inaccurate interpolation" for each technique for the **EUS** problem.

Samples /Rank	CGAL Serial (s)	Interpolation (s)	Communication (s)
7.2M	178	0.00246	
2.1M	53	0.00141	0.00125
887k	21	0.00093	

Table 13. Distributed memory *post hoc* interpolation cost for 100,000 particles across a **single interval** of the Cloverleaf3D extracted data using 16 compute nodes and 96 MPI ranks on Summit. Values averaged over all reconstruction runs.

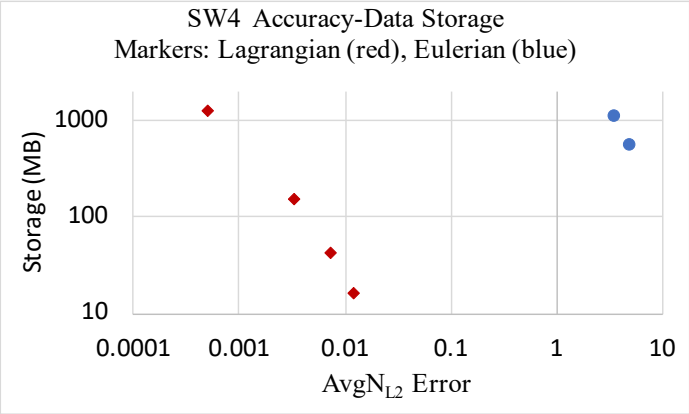


Figure 39. SW4 accuracy-data storage scatter plot.

For **PHE-2**, we measured the time required for Cloverleaf3D reconstructions. In our empirical study, we only reconstructed Cloverleaf3D pathlines in a distributed-memory setting (16 nodes, 96 MPI ranks). Table 13 contains timings for our reconstruction method for a single interval given a workload, i.e., number of samples to be triangulated and interpolated per rank. The most dominant cost during this process is the search structure construction, i.e., the Delaunay triangulation. Although we avoid the prohibitive cost of a global Delaunay triangulation with our implementation, we believe there is room for improvement. That being said, for reduced Lagrangian representations, the parallel Delaunay construction cost can be comparable to the Eulerian approach that requires performing interpolation and communication for every cycle. For example, the Lagrangian configuration using an interval of 40 and 1:64 number of particles, can be used to construct pathlines for 100,000 particles across 600 cycles in under 6 minutes (excluding I/O). For an Eulerian approach to be faster, it would need to compute each cycle in 0.6 seconds (our implementation required 0.47 seconds per cycle excluding I/O; approximately 5 minutes).

7.5.2.2 SW4 Seismic Wave Propagation Simulation.

For the SW4 simulation, we considered 4 options for number of particles. The SW4 simulation generates a displacement vector field that captures the wave propagation modeled in the simulation. In this case, the Lagrangian representation is far better equipped than the traditional method to accurately encode this transient behavior in the domain. Our experiments considered 2000 cycles of the simulation, and evaluated accuracy by reconstructing 90,000 test particle trajectories placed randomly between Z=5,000 and Z=15,000 (the layer of most activity in the domain) using a step size of 1.

The SW4 simulation domain extents are very large, thus each cell side is approximately 387 in our experiments. The $\mathbf{AvgN}_{L2}$ value of our test particles indicates all particles remained within the cell, with the Lagrangian technique offering near perfect reconstruction, while the Eulerian technique only suffers from an error of 1% of the cell side by this measure. However, displacement values in an earthquake simulation are expected to be small and an error of even that magnitude might represent failure to capture the wave propagation.

The SW4 histogram plots (Figure 43) use different bin ranges for Lagrangian and Eulerian given the distributions were very different. The plots show the an increase in error for the Eulerian technique as the interval increases and an increase in error for the Lagrangian technique as the number of particles used decreases from 1:27 to 1:64.

Overall, the Lagrangian technique offers excellent propositions for **DS-1** and **PHE-1**. The Lagrangian technique was able to preserve near perfect integrity with upto 70X less data storage.

7.5.2.3 Nyx Cosmology Simulation.

For the Nyx cosmology simulation, we considered 4 options for number of particles and interval to provide an understanding across a wider spatiotemporal range. Figure 40 shows a slice of the Nyx vector field at two three slices (0, 200, 400). We observed that the unit vectors at each grid point in the domain remain relatively the same across all cycles. The slow evolution of the vector field is in terms of velocity magnitude in few regions of the domain. The maximum velocity magnitude in the domain increases steadily for the 400 cycles of the simulation we use in this study. Our experiments considered 50,000 test particle trajectories placed randomly in the domain and set a step size to 0.02.

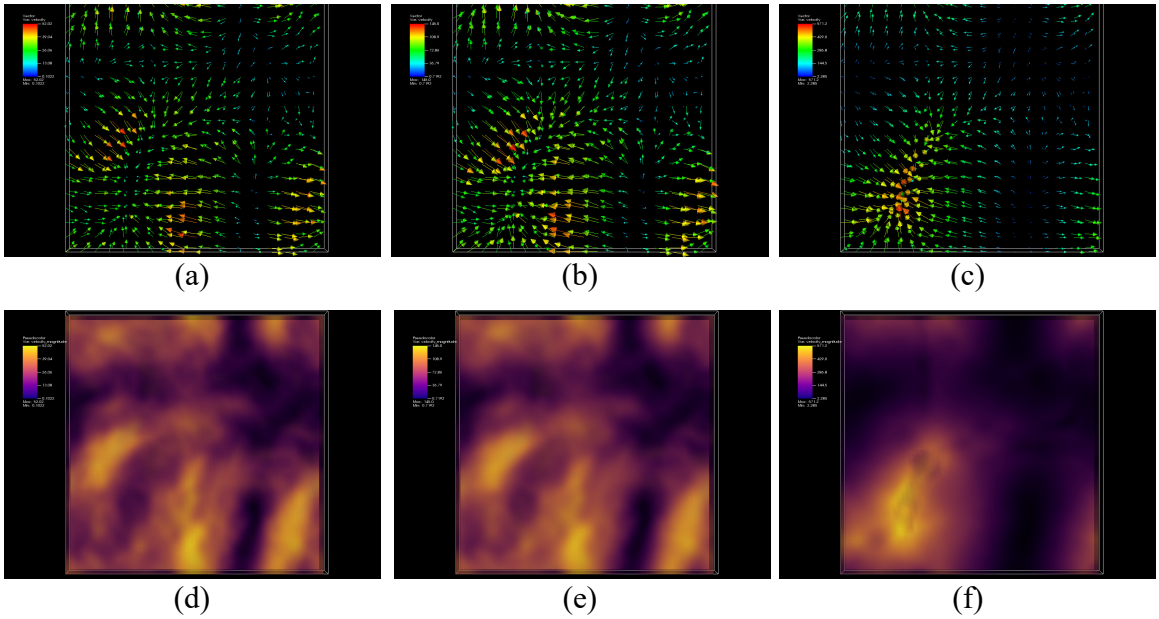


Figure 40. Nyx vector field visualization: (a) and (d) show the vector field at time 0 and the maximum velocity magnitude is 52.02, (b) and (e) show the vector field at time 200 and the maximum velocity magnitude is 145.0, and finally, (c) and (f) show the vector field at time 400 and the maximum velocity is 571.2.

An interesting outcome of these experiments was observing **PHE-1**, i.e., accuracy, given the variation in interval. The Eulerian technique is nearly

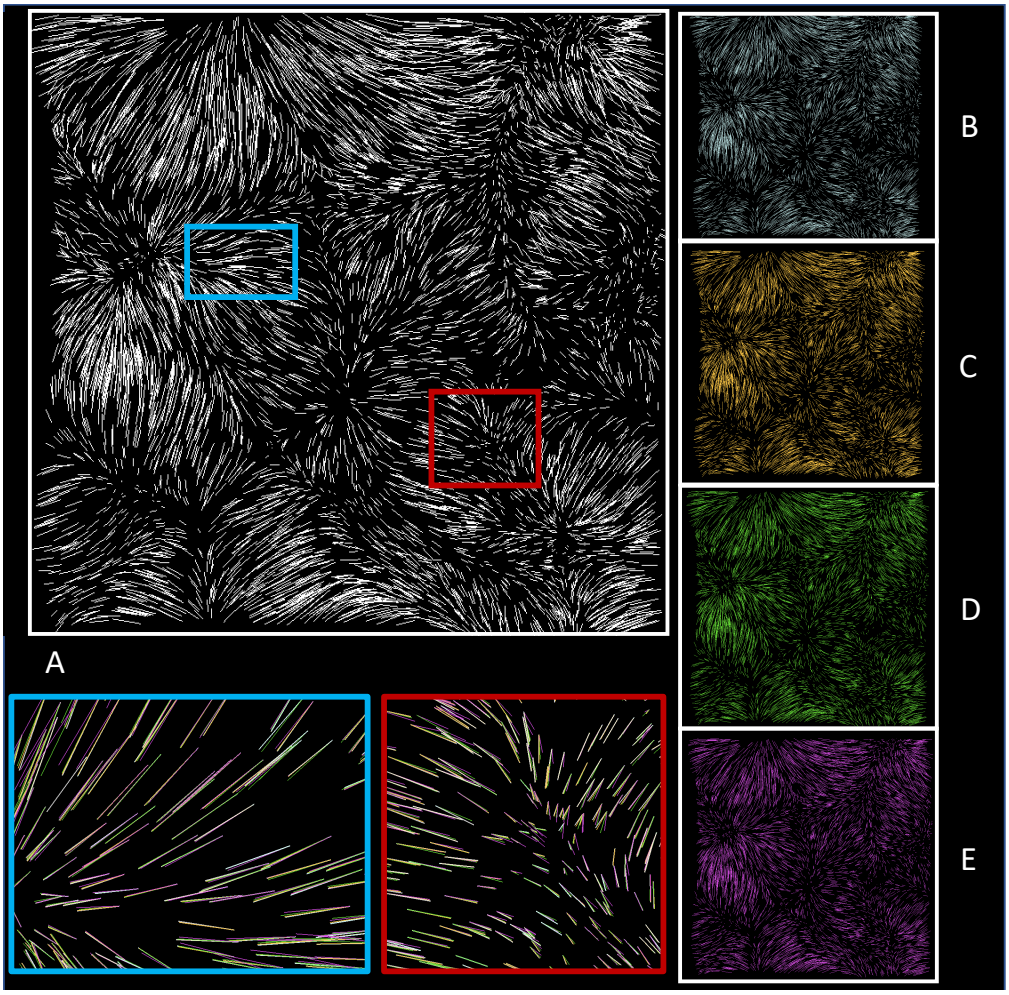


Figure 41. A qualitative comparison of pathline reconstruction over one interval for the Nyx data (**Interval = 25** case). For each configuration we specify **DS-1** and **PHE-1** as (Bytes, Cell Side%). Image A (white) is the ground truth, B (light-blue) is Eulerian (227MB; 2.29%), C (yellow) is Lagrangian 1:1 (232MB; 11.61%), D (green) is Lagrangian 1:8 (27MB; 37.27%), E (pink) is Lagrangian 1:27 (8MB; 72.727%).

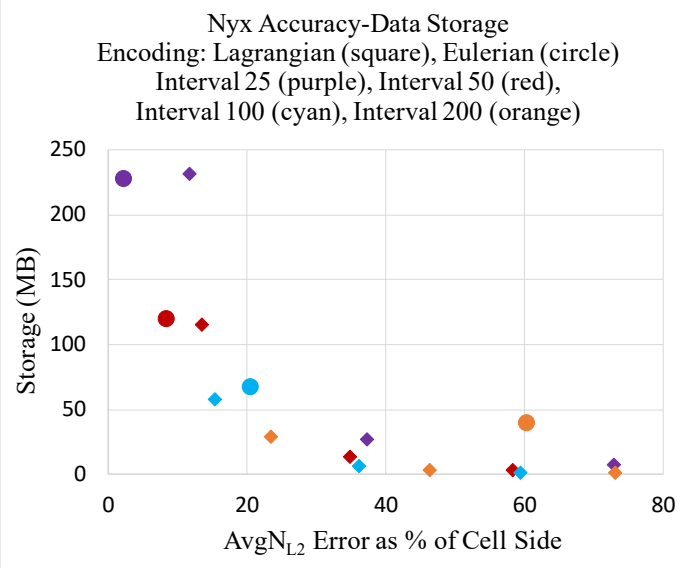


Figure 42. Nyx accuracy-storage scatter plot.

perfectly accurate when the sampling is less sparse (interval = 25). In contrast, the Lagrangian technique is less accurate, even when using the same number of particles as grid points. Such behavior is expected when the variation between the vector field across cycles is very small (Figure 40). In such a setting, using only a fourth-order Runge Kutta (RK4) provides more accurate interpolation than applying a second-order barycentric coordinates interpolation on top of the trajectories extracted using RK4. This "stitching" error has been studied in several prior works [32, 13, 14, 8].

As the interval size increases, i.e., the **S**parsity component of the **EUS** problem, we observe Lagrangian improving in accuracy and offering multiple favorable data storage-accuracy propositions. For example, for an interval of 200, greater accuracy can be achieved by the Lagrangian technique using a 10X data reduction. The behavior of techniques (one losing integrity as sparsity increases; another becoming more accurate as sparsity increases) is well captured by the histograms in Figure 44. Of course, a longer interval does not guarantee better

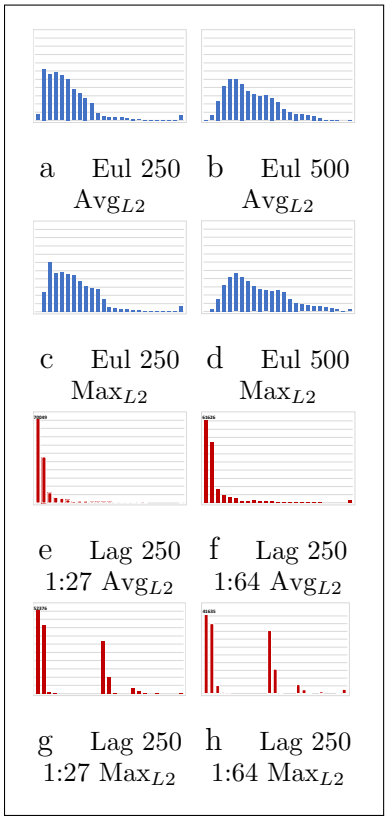


Figure 43. **SW4** experiment histograms for 90,000 test particle interpolation errors. Each plot has 25 bins, Eulerian bins range from <0.6 to >15, Lagrangian bins range from 0 to >0.2, with bar height encoding number of particles. Horizontal grid lines mark increments of 2,000.

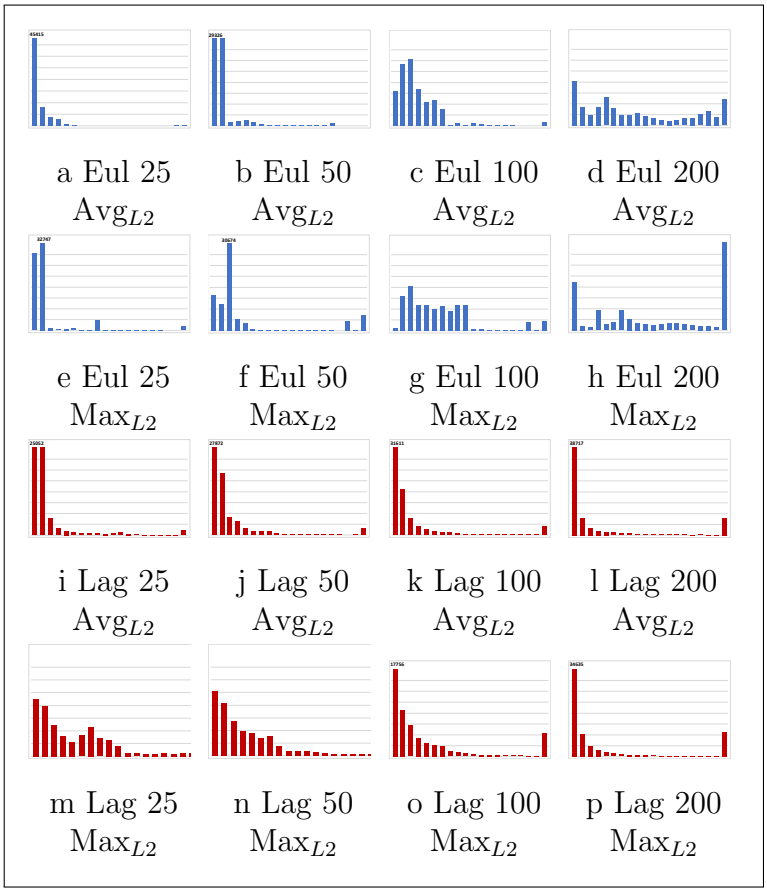


Figure 44. **Nyx** experiment histograms for 50,000 test particle interpolation errors. Each plot has 20 bins, ranging from 0 to >0.44, with bar height encoding number of particles. Horizontal grid lines mark increments of 2,000.

accuracy for the Lagrangian technique in all settings. Divergence over a long interval impacts Lagrangian-based interpolation accuracy [34].

With respect to the impact of number of particles on **PHE-1** and **DS-1**, we observe that error increases due to data reductions are higher when the interval is smaller. For example, error increases by 3X for a 27X data reduction for interval 200 and error increases by over 6X for a 27X data reduction for interval 25. We note that in each of these cases, the increases in error resulted in $\mathbf{AvgN}_{L2}$ values that still indicate that the majority of test particles were within the same cell as the ground truth particle. This result supports the notion that the Lagrangian technique can effectively support data reduction in settings of temporal sparsity. Overall, our empirical study using the derived Nyx vector field produced interesting trends that support the use of Lagrangian technique for **EUS** problem, while also demonstrating a stitching error when storing data to disk more frequently.

7.6 Conclusion

We contribute an empirical study in response to uncertainty regarding whether or not the **L-ISR-PHE** workflow is practically viable and should be the preferred solution for the **EUS** setting. Although previous works had demonstrated compelling propositions with respect to accuracy-storage tradeoffs, they had been mostly performed in theoretical *in situ* environments. This research gap concerning practical *in situ* encumbrance and viability on a supercomputer is a barrier for adoption. Filling this research gap is the key contribution of our empirical study. We provide insight on this front by considering execution time, memory usage, and percentage of time spent by the simulation on *in situ* processing. Our key findings show that simulations almost always spent less than 10% of time on *in situ* processing and in some cases, less than 1%. For the *post hoc* phase, our

empirical study improves on prior evaluations of data storage-accuracy propositions both quantitatively and qualitatively. We present per particle outcomes using histograms and believe this representation accurately captures interpolation error changes across configurations. For **EUS** settings, our experiments demonstrate significant data storage reduction (8X-200X) while maintaining accuracy (in every case, particles remained within ground truth cell on average). Further, we provide cost estimates for a Lagrangian-based distributed-memory *post hoc* advection scheme. Overall, we believe this empirical study addresses the existing research gap concerning *in situ* encumbrance on a supercomputer and contributes to existing evaluations of *post hoc* efficacy.

Part 3

Conclusion and Future Work

This part concludes the dissertation. Chapter VIII provides a synthesis of the findings and recommendations for future work. Chapter IX comes from the work done as part of my oral comprehensive exam. **L-ISR-PHE** is yet in the early stages of its development and use as a data reduction operator. The task of selecting the best set of data during in situ Lagrangian extraction, however, has parallels to existing research. Chapter IX presents a survey of seed placement and streamline selection algorithms. These are a large number of algorithms that have been proposed over the past two decades to calculate representative integral curves in a vector field. Although these works have been predominantly proposed in a post hoc processing context, this problem is very closely associated given the commonality of the underlying problem. Overall, several of the studies in Chapter IX could inspire new in situ Lagrangian representation extraction techniques.

CHAPTER VIII

CONCLUSION

8.1 Synthesis

The ability to perform accurate exploratory time-dependent vector field visualization is threatened by I/O gap trends seen at exascale (and expected to continue beyond). The **L-ISR-PHE** is a viable solution to that problem. This dissertation identified research gaps pertaining to in situ encumbrance, accuracy-performance tradeoffs, and value/challenges of advanced techniques. Research conducted in this dissertation investigates these research gaps and provides answers for the corresponding research questions listed in Chapter I.

- **RQ1.1 What is the practical in situ encumbrance of the technique?** Chapter VII presents a study that finds in situ Lagrangian representation extraction can in most cases (high and low simulation cycle time) be performed within in situ processing resource budgets, i.e., between 10% to 20% of the simulation time. We found that in most cases, our implementation cost less than 10% and in some cases, as low as under 1%.

- **RQ1.2 Are there ways to reduce in situ encumbrance?** Chapter VI evaluates a communication-free model and finds that reconstruction accuracy is high for short and medium length integration intervals. A communication-free model offers an accuracy-performance tradeoff that provides good propositions in many practical settings. Further, Chapter VII showed that multi-core CPUs offer fast in situ computation of particle trajectories. Hybrid CPU-GPU techniques that could maximize resource utilization could further reduce in situ encumbrance.

– **RQ2.1 Are results maintained over varying data sets? Especially non-analytical data sets?** Yes. This dissertation considered multiple data sets across all studies and observed consistent trends related to accuracy-storage propositions.

– **RQ2.2 Is this improvement necessary when considering absolute error?** We understand this is use case/application dependent. That being said, as temporal sparsity increases, the Lagrangian technique is far more accurate. Further, this dissertation includes a qualitative evaluation that highlights the benefits of using a Lagrangian representation compared to an Eulerian representation. Overall, we believe the improvement is necessary given the benefits of reduced data size and improved or maintained accuracy.

– **RQ2.3 What is the spectrum of outcomes compared to the traditional approach?** Chapter VII extends the evaluation of **L-ISR-PHE** by considering the outcomes on a per particle basis using histograms. The use of these histograms enables a comparison to the traditional approach across several configurations and provides insight into reconstruction error distribution and its relation to temporal sparsity.

– **RQ3.1 Can we benefit by innovating past the simple scheme?** Chapter V explored novel techniques and introduce variable duration variable placement schemes with the goal to reduce reconstruction error. The study found improved accuracy-storage propositions across all data sets and demonstrated the benefits of more advanced schemes.

– **RQ3.2 What new challenges are created by these advancements and can they be addressed?** More advanced schemes introduce complexity

due to the need of a global solution in many cases (for example, a global triangulation) and by likely introducing an unstructured point set. However, in many cases local solutions can be computed in a distributed environment and offer speed ups.

Overall, our findings show that **L-ISR-PHE** is a viable and effective alternative for the **EUS** problem to traditional approaches at scale. Computing local flow maps and using Lagrangian representation that utilize longer trajectory forms advance the effectiveness of the technique either from a performance and accuracy standpoint, respectively. More advanced techniques combining these methods and considering boundary conditions would benefit from gains in both accuracy and performance. Overall, we addressed each research gap and can answer the central dissertation question as "yes."

8.2 Future Work and Research Directions

Given in situ Lagrangian analysis is a relatively new research area, the foundational work that is part of this dissertation has identified several key areas of research going forward. We break our discussion of future research works into two topics: in situ sampling strategies and post hoc reconstruction techniques.

8.2.1 In Situ Sampling Strategies. The general in situ Lagrangian analysis framework offers plenty of flexibility with respect to spatial and temporal sampling strategies. Future researchers can evaluate the costs of using more computation time to execute improved sampling strategies or the use of more memory to store more state information about individual samples. With respect to spatial sampling, feature-guided techniques and machine learning offer solutions to identify an optimal set of Lagrangian basis flows to store. However, identifying scalable techniques that improve information content per byte stored to disk is the

challenge. Further, it would be valuable to see in situ Lagrangian analysis deployed for an application and configured to accurately capture an application-specific feature.

In the area of temporal sampling, there are plenty of open challenges with respect to applying curve simplification strategies to pathline trajectories in situ and identifying the best set of data points to store to disk. Evaluation along the temporal axis will require exploring strategies to manage high in situ memory consumption patterns. The successful development of efficient techniques could potentially result in the accurate reconstruction of the entire temporal resolution of a time-dependent vector field.

8.2.2 Post Hoc Reconstruction Techniques. Although the focus of this dissertation was demonstrating the viability and efficacy of extracting a Lagrangian representation of a time-dependent vector field, several of our studies improved and advanced the state of the art with respect to Lagrangian-based advection schemes. In the future, development of fast and scalable interpolation methods would enable interactive time-dependent vector field visualization. Additionally, there is the need to incorporate Lagrangian-based advection schemes into vector field visualization frameworks.

Other research works in the space of post hoc reconstruction techniques can consider methods to improve reconstruction quality at the boundary of nodes or identification of optimal Lagrangian basis flow neighborhoods at each step. In general, however, post hoc reconstruction methods are closely tied to their in situ extraction counterparts — decisions made during extraction, can offer require special treatment during the post hoc reconstruction.

8.2.3 Extending Evaluations.

The flow visualization community would benefit from an extensive evaluation (quantitative and qualitative) of the effects of increasing temporal sparsity on traditional exploratory flow visualization integrity. Another valuable future direction of research is to extend evaluations and consider other data reduction mechanisms that could be applied for vector fields.

CHAPTER IX

SURVEY OF SEED PLACEMENT AND STREAMLINE SELECTION TECHNIQUES

Most of the text in this chapter comes from a publication [10], which was a collaboration between Roxana Bujack, Christoph Garth, Hank Childs, and myself. I was the primary contributor of this work and I was responsible for surveying the field, creating a classification, and writing the manuscript. Roxana Bujack, Christoph Garth, and Hank Childs provided timely, valuable feedback and were involved in editing the manuscript.

In this chapter, we survey an extensively researched problem: how to place seeds and select integral curves (the majority of the literature focuses on streamlines) such that a representative visualization of the vector field can be produced. The underlying commonality of the problem to that of sampling during in situ Lagrangian analysis, makes this a very relevant and foundational field of research for future in situ Lagrangian analysis techniques.

9.1 Introduction

Advances in parallelization technology have enabled efficient computation of a large numbers of streamlines [79, 60]. Using a large number of streamlines, however, does not guarantee a useful visualization. Thus, motivated by the need to assist scientists with the exploration of flow fields in various contexts (planar surface, curved surface, or volume flow), the identification of initial seed placement and the selection of streamlines to visualize has been an active research area. In particular, several research efforts have aimed to automatically generate or select a representative set of streamlines for a given flow field. A survey of these seed placement and streamline selection (SPSS) techniques is the focus of this chapter.

Concerning previous work, the survey by McLoughlin et al. [80] is the most notable study that has addressed seed placement for flow visualization. Although covering the most prominent studies up until that time, SPSS techniques were not a central theme of the study, and the techniques used to identify a representative set of streamlines have significantly evolved since. Over the past decade, several studies have presented techniques that identify representative streamlines via selection from a random set as opposed to iterative generation methods (typically seen in seed placement algorithms). Further, on the topics of general feature extraction or vector field clustering techniques (often employed in SPSS workflows), studies by Post et al. [81], Laramee et al. [82, 83], and Pobitzer et al. [84] provide comprehensive coverage.

This survey is organized as follows: Section 9.2 covers background information on SPSS, including the mathematical definition of a streamline, challenges, desired characteristics of visualizations, and evaluation methodology. Sections 9.3 introduces a high-level classification of techniques and a basic road map for the reader to navigate the survey. Sections 9.4 and 9.5 contain details of techniques for each technique and strategy. Section 9.6 discusses future work, and finally, Section 9.7 highlights our contributions and concludes the report.

9.2 Seed Placement and Streamline Selection Background

This section discusses key aspects of the seed placement and streamline selection (SPSS) techniques. First, we differentiate between seed placement and streamline selection from a terminology point of view. A seed placement algorithm is the process of selecting particle seed locations to calculate useful streamlines. In contrast, a streamline selection algorithm is the process of choosing useful streamlines from a large set of precalculated streamlines (typically generated

using a random seed placement). In several instances, both seed placement and streamline selection are used together to produce the desired outcome.

The following sections cover the challenges and application contexts, desired characteristics of visualizations, and the evaluation methodologies used with respect to SPSS techniques. Further, we introduce the axes we use in this survey to evaluate different classes of SPSS techniques.

9.2.1 Streamlines and Pathlines. A streamline is a curve $x_s : \mathbb{R} \to \mathbb{R}^d$ that is everywhere tangential to the instantaneous velocity of an unsteady vector field $v : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ at one fixed time t

$$\frac{dx_s(s)}{ds} = v(x_s(s), t), \quad x_s(s) = x_s(s_0) + \int_{s_0}^{s} v(x_s(\tau), t)d\tau \tag{9.1}$$

while a pathline $x_p : \mathbb{R} \to \mathbb{R}^d$ is tangential to the vector field over time

$$\frac{dx_p(t)}{dt} = v(x_p(t), t), \quad x_p(t) = x_p(t_0) + \int_{t_0}^{t} v(x_p(\tau), \tau)d\tau. \tag{9.2}$$

Both are uniquely defined through a differentiable vector field and the location and time of their seedpoints $x_s(s_0), x_p(t_0)$ [85].

9.2.2 Challenges. SPSS strategies have been proposed to address various flow visualization tasks. In this survey, the majority of research studies propose an algorithm to *generate informative flow visualizations using a representative set of streamlines.*

The wide usage of streamlines to effectively visualize a flow field has resulted in several research efforts directed at the generation of a representative set of streamlines. Producing an image using an excessively large number of streamlines can result in a dense and cluttered visualization showing redundant information. However, if the number of streamlines used is too few, important flow features can be missed. Further, in a three-dimensional settings, streamlines often occlude one another. It is desirable that less informative streamlines do not occlude streamlines

capturing important features of the flow. Streamline length is an additional consideration when proposing an algorithm. Uniformly placed short streamlines often result in visual artifacts, whereas complete streamlines (i.e., streamlines that only terminate at boundaries or critical points) might result in cluttered non-uniform distributions.

Although comprising a smaller body of work, we additionally consider SPSS techniques for flow visualization tasks such as:

- Selection of streamlines that are similar to a query streamline, i.e., a user-specified streamline.
- Particle-based flow visualization systems that require seed placement to control particle density distribution.

Our survey includes these studies, given the commonality of the underlying objectives and algorithms. These studies are useful in that they address challenges like identifying similar streamlines in an orientation, position, and scale invariant manner, particle density distribution management, and minimization of vector field reconstruction error.

9.2.3 Desired Characteristics. Verma et al. [15] were the first to explicitly list characteristics that are desired of the selected set of representative streamlines. These characteristics are:

- **Coverage:** Streamlines should not miss interesting regions of the flow field.
- **Uniformity:** Streamlines should be uniformly distributed over the field.
- **Continuity:** From an aesthetic perspective, streamlines should be selected such that they show continuity in the flow, i.e., long streamlines are preferred.

However, these desired characteristics have been modified by researchers and scientists as this area of research has evolved, particularly after considering 3D volume flows, view of the domain, specific features of interest, and information content. Thus, additionally desired characteristics include:

- Visibility of regions of interest (ROI), i.e., occlusion management [86].
- Smooth transitions or frame coherence when visualizing time-dependent flow or changing viewpoints [87, 88]
- Retaining spatial perception for depth cues [89]
- Representing maximum information content using the least number of streamlines [90]

Contributions to this area of research have prioritized different characteristics while advancing or improving on previous work — either from a visualization or computational perspective.

9.2.4 Evaluation Methodology. The results of SPSS techniques have seen on-going improvements over the past two decades. These improvements have been evaluated qualitatively and/or quantitatively. The majority of these visualization research studies have used a qualitative evaluation to demonstrate that the proposed technique achieves some desired characteristic better than a previous approach. Although fewer in number, multiple studies have quantitatively evaluated a technique by considering the accuracy of vector field reconstruction using the selected streamlines or computational performance.

Qualitative evaluations can be biased based on the specific requirements or objectives of an individual study. Thus, capabilities such as maintaining spatial perception, or highlighting multiple ROI are viewed as "upgrades." To limit the

scope of this survey, rendering techniques, such as thinning of lines or lighting effects, are not discussed. That being said, choices surrounding the rendering and presentation of streamlines can contribute to and improve our perception and understanding of the flow field.

In this survey, we evaluate classes of strategies based on three factors. These three factors enable our analysis of techniques, and we incorporate the qualitative and quantitative criteria determined by the studies themselves. These three factors are:

- **Regions of interest (ROI):** Evaluation of whether a technique can identify and focus on ROI.
- **Minimization of redundancy:** Evaluation of whether a technique mitigates the selection of redundant streamlines.
- **Computational performance:** Evaluation of whether a technique can be used in computationally constrained contexts.

The ROI axis is important to evaluate the ability of an algorithm to primarily focus on salient features, avoid occlusion by less important streamlines in 3D, and generate a visualization that naturally draws the user's focus to the ROI of the field [15, 86, 91]. The redundancy axis is important to evaluate whether an algorithm is selecting multiple streamlines that convey relatively the same information (e.g., parallel streamlines) [92, 90]. Considering redundancy is particularly useful when there is a tradeoff between the number of streamlines that can be used and the total information conveyed by the set of streamlines. The computation axis is useful to understand the cost of a particular strategy and its viability under different scenarios (e.g., interactive, in situ, distributed memory). In

total, considering these three axes informs recommendations for which technique to use depending on the application and constraints.

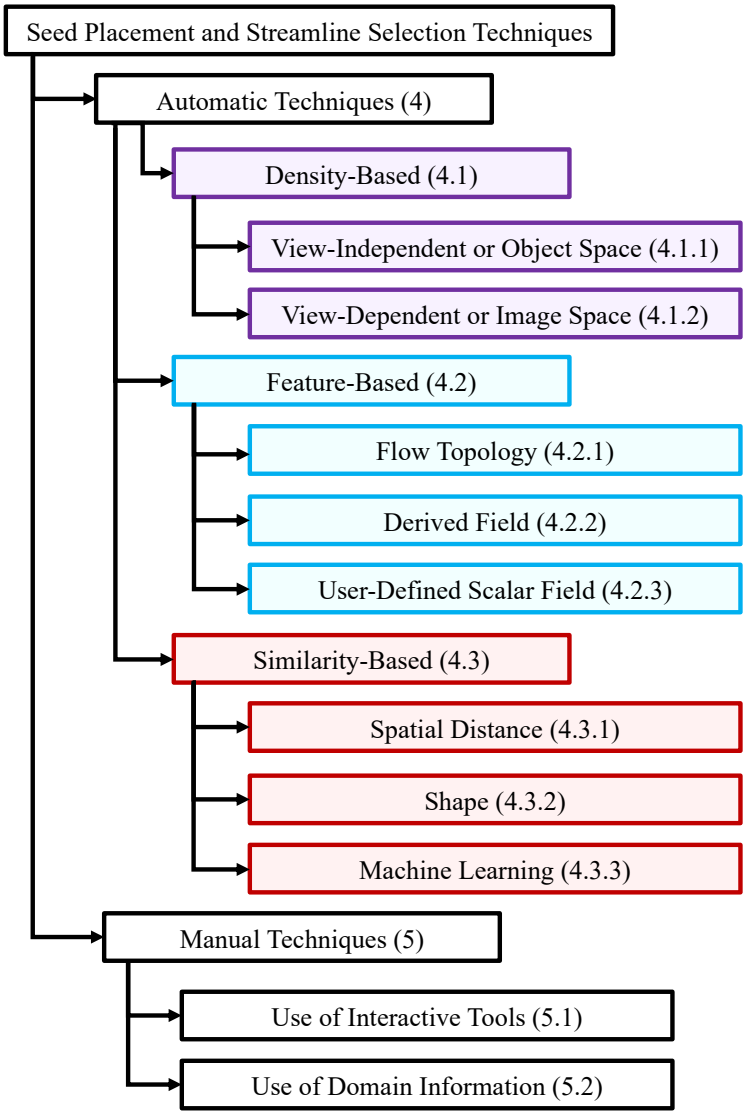


Figure 45. Classification tree for seed placement and streamline selection techniques. To assist with correlating this classification with Figure 46, we color subclasses of density-based purple, feature-based blue, and similarity-based red. Finally, each subclass has its corresponding subsection within the survey listed parenthetically.

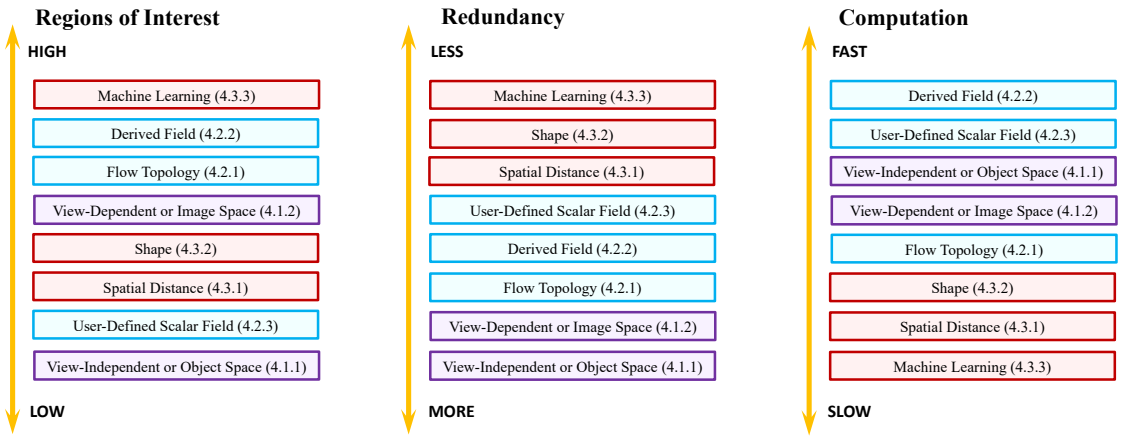


Figure 46. Approximate, general ordering of the identified categories of seed placement and streamline selection techniques based on three evaluation axes, as discussed in Section 9.3. We use the same color coding as used in Figure 45. We intend on the rating along the evaluation axes to be used as an approximate guide for the reader to identify categories of interest based on constraints or application contexts.

9.3 Classification

To explore a flow field, without any assistance or prior knowledge, a scientist would be required to select locations for seed placement, followed by the generation of streamlines. Based on the computed visualization, the scientist can then iteratively refine their seed placement to produce a meaningful visualization. However, this method can be challenging when dealing with complex flow fields. To assist scientists with this challenge, two high-level classes of approaches have been developed over the past two decades. The first class of techniques is automatic, i.e., researchers have automated the process by developing algorithms that produce a set of streamlines that convey flow field information. The second class of techniques is manual, i.e., researchers employ interactive methods or leverage domain knowledge to place seed points and manage streamlines. Given elements of

overlap between several of the proposed algorithms, we aim to classify studies based on the technique contributed to generate or select streamlines.

As shown in Figure 45, we further classify automatic techniques into density-based (purple), feature-based (blue), and similarity-based (red) strategies. Within each of these classes, we identify subclasses of techniques. The manual techniques surveyed are classified into interactive tools for the placement or control of streamlines and strategies that use domain-information for seed placement.

Automatic techniques have significant diversity in strategy. Figure 46 shows an evaluation of automatic techniques along three axes (introduced in Section 9.2.4). Given that each class of algorithms contains subclasses and multiple works, our rating is approximate and relatively general. We base our rating using comparisons (both qualitative and quantitative) made within research works themselves, type of algorithms, scalability of the solution, and our overall understanding of the field. The majority of research studies conduct comparative evaluations with previous work qualitatively and/or quantitatively (e.g., [93, 89, 94] compare against multiple other works). These ratings have not been established by conducting new experiments, and certain works within subcategories are exceptions to the position of the entire subcategory along our axis. The objective of the figure is to provide an approximate guide for the reader to navigate the large space of SPSS techniques. Further, the difference between categories might not necessarily be as vast as the position along an axis might suggest. For example, on the ROI axis, although a feature-based strategy will capture a ROI better than a view-independent technique and is rated higher, the view-independent technique will indeed have sampled the ROI albeit without any specific focus.

If the reader desires to find a strategy for their own SPSS problem, this evaluation can be used as an approximate guide. For example, assume the reader is interested in a high ability to capture ROI, is indifferent toward redundancy, and desires fast computation, then the reader might be interested in exploring the *Derived Field* class of strategies.

Lastly, although we do not organize this survey based on the target contexts of SPSS studies, we believe such a grouping is valuable. Readers can reference Table 16 to identify works which consider a specific target (for example, steady state volume flow).

9.4 Automatic Techniques

Automatic seed placement or streamline selection algorithms follow a set of rules to generate a distribution/selection of streamlines (or particles in some cases). These algorithms may consider the view direction, properties of the vector field, properties of integrated streamlines and so on. Our classification identifies whether a particular algorithm is primarily contributing a density-based (9.4.1), feature-based (9.4.2), or similarity-based (9.4.3) approach. For example, an algorithm might first extract flow feature locations and strategically place seed points in ROI before placing additional seeds to generate an approximately uniform distribution of streamlines while highlighting flow features — we categorize this as a feature-based approach and not a density-based approach. Often, the motivation of studies will overlap given the desired characteristics are not mutually exclusive, i.e, an algorithm may strive to achieve several desired characteristics in some order of priority.

9.4.1 Density-Based. Density-based techniques are typically proposed when a uniform or user-defined distribution of streamlines (not seed

points) is the desired outcome of an SPSS algorithm. A uniform distribution of streamlines provides the user with an overview of the entire flow field. Corresponding techniques typically generate approximately evenly-spaced streamlines in object or image space. We categorize the density-based approaches as view-independent (9.4.1.1) or view-dependent (9.4.1.2). For view-independent (object space) approaches, the resultant set of streamlines do not change if the view of the domain changes. Whereas, view-dependent (image space) techniques might select different sets of streamlines when the viewpoint of the domain changes.

9.4.1.1 View-Independent or Object Space Techniques.

Density-based view-independent approaches propose algorithms to obtain a uniform or user-defined density distribution of streamlines in object space. The remainder of this section is divided into algorithms that use local or global seeding strategies.

Algorithms Using Local Seeding Strategy

The first use of an automatic seed placement technique to maintain distances between streamlines was by Hultquist et al. [95]. Hultquist et al.'s early work considered seed point addition and removal in the context of stream surface construction. After seed points are initialized as a rake, the distance between particles is tracked as particle trajectories (streamlines) are integrated. Based on the premise that to achieve a good surface visualization an approximately uniform spacing between streamlines is desired, new seeds are added or existing seeds are merged based on a user-defined neighboring particle distance criterion.

Max et al. [96] used evenly-spaced short streamlines to visualize a 3D vector field on a contour surface. They considered several projections to visualize the streamlines. While they evaluated different projections (Eye, Normal, XY,

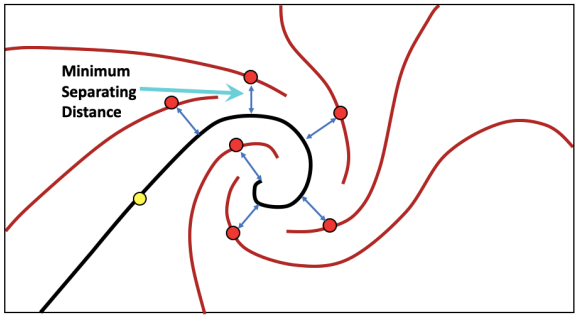


Figure 47. Candidate seed points are identified at locations a minimum separating distance away from the initial streamline (thick). Image inspired by Jobard et al. [97].

and Cylinder) on the 3D surface and transitions of those projections as the view changes, a precomputation phase involved seed placement and particle tracing in object space. To allow streamlines to be traced for long distances before they get too close to each other, the initial positions of seed points are chosen on an integer lattice in a spatially hierarchical manner. A streamline length threshold is used to determine the minimal length of accepted streamlines. A streamline grows until it reaches a surface edge, a singularity in the field, or becomes too close to another streamline.

Jobard and Lefer extended the work done by Max et al. [96] and proposed an effective and popular single pass method for placement of long evenly-spaced streamlines in a 2D steady state field [97]. The method can achieve visualizations ranging from dense texture-like to sparse hand-drawing styles by only setting the separating distance, denoted by d_{sep}, between adjacent streamlines. The algorithm initially places a random seed point and integrates a new streamline backward and forward until some termination criterion is met. The first streamline is used to calculate a set of candidate seed points d_{sep} distance away from the streamline. The candidate seed points are added to a queue to be evaluated. Each candidate seed

point is used as a starting location to integrate a streamline until it is within some distance d_{test} (a fraction of d_{sep}) from existing streamlines. Figure 47 illustrates seed points, a user-defined distance away from an initial streamline, used to integrate new streamlines. If the integrated streamline is accepted, then the new streamline contributes a set of candidate seed points to the existing queue. To accelerate the computation process, Jobard and Lefer proposed two optimizations:

- Streamlines consist of a set of evenly-spaced sample points that are a distance smaller than d_{sep} apart. Only these sample points are considered in distance computations.

- A cartesian grid with cell side d_{sep} is superposed on the domain to support binning of sample points and limit distance computations to surrounding cells.

These optimizations have been employed in several following research works.
The algorithm achieved placement quality as good as previous techniques, i.e., work by Turk and Banks [98] (described in Section 9.4.1.2), while significantly improving computation speeds. Jobard and Lefer extended their initial work to propose a multiresolution technique for steady state flow [99] and an approach to create animations for visualizing unsteady flow [100]. To generate a sequence of streamline-based images of a vector field with different density (multiresolution), they computed an initial set of streamlines for a large separating distance value. The resulting streamlines then form an initial set of streamlines for the next level, i.e., an image that has a higher streamline density and uses a smaller separating distance value. This process is performed for the desired number of levels of streamline density. The shortcoming of this approach is that streamlines traced for later levels were shorter due to the existence of the initial set of longer streamlines.

For the visualization of unsteady flow in 2D, they proposed a feed-forward algorithm that used *reference* streamlines from one time step to select *corresponding* streamlines in the next time step. Sample points of *reference* streamlines act as initial seed locations to generate candidate streamlines. The best candidate streamline, based on an L2-norm correlation measure, is selected as a corresponding streamline in the next time step. If required, additional streamlines are calculated to obtain a uniform distribution. By correlating instantaneous visualizations of the vector field at the streamline level, they animated 2D unsteady flow visualization.

Mattausch et al. [101] adopted Jobard's and Lefer's algorithm [97] with the aim to improve focus+context techniques and the spatial perception of 3D flow fields. To extend to 3D, six candidate seed points are calculated at a distance d_{sep} for every sample point on a streamline. Additionally, they improved the multiresolution technique presented by Jobard and Lefer [99] by preventing the generation of shorter streamlines for higher levels of detail.

Jobard's and Lefer's algorithm has been utilized as an intermediate step for texture-based flow visualization techniques and domains such as DTI Fiber Tracking [102, 103]. Li et al. [104] presented Chameleon, a texture-based rendering framework, which decouples the calculation of streamlines and the mapping of visual attributes, allowing flexible control of the visual appearance of the vector field. The seed placement algorithm is employed to control the length and density of the generated streamlines. A trace volume is created using a dense set of evenly-spaced streamlines and their geometric properties. The trace volume can then be combined with varying input appearance textures to produce a wide range of effects interactively. Shen et al. [105] extended the Chameleon framework to support

unsteady flow fields by calculating pathlines instead of streamlines. The algorithm tracked pathline segment intersections and trace volume updates during rendering. However, the study did not address the distribution of pathlines across the domain over time.

Employing a pipeline similar to Chameleon, Helgeland et al. [106] proposed a method to use evenly-distributed particles as input for a texture-based visualization of unsteady flow in 3D. The algorithm outputs a point set, i.e,, seed locations, instead of a set of streamlines. Using an initially random pool of seed points, Jobard's and Lefer's algorithm [97] is applied to identify the subset of seed points that generate a set of streamlines d_{sep} distance apart. The resultant point set is used to generate streamlines using a texture-based method (for example, Seed LIC [107]) for each time step. After each advection step, cluttering is avoided by removing particles that are less than d_{test} distance apart. While particles leaving the domain are naturally removed, particles are added to account for inflow. A seed point is added to the center of boundary voxels if a fixed length streamline traced from it is d_{sep} distance from existing streamlines. Overall, particle density is maintained by injecting particles into areas with low density without exceeding a user-defined maximum number of seed points for the domain.

Algorithms Using Global Seeding Strategy

Unlike local seeding strategies that place seeds in the vicinity of previously placed streamlines, Mebarki et al. [39] proposed to place seeds furthest away from all previously placed streamlines. Using an approach proposed by Chew et al. [108] that had already been successfully applied to point sampling and mesh refinement [109, 110, 111], Mebarki et al. place new seed points at the center of the biggest voids within the domain. Using Delaunay triangulation to identify voids

in the domain, the circumcenter of the triangle with the largest circumradius is chosen as the next seed location. Streamlines, approximated using a set of sample points, are inserted one at a time and are traced until a minimum separating distance criterion is violated. Processing a priority queue of triangles, sorted by circumradius and with circumcircle diameter larger than the separating distance, the algorithm ends when the priority queue is empty. The computation of the process is significantly optimized by only using every n^{th} sample point to calculate the Delaunay triangulation and only adding triangles incident to the streamline end points to the priority queue. Placing seed points farthest away from existing streamlines results in long streamlines, improving on the quality of representative streamlines by reducing streamline discontinuities. Mebarki et al. demonstrated reduced execution time compared to Jobard and Lefer [97] for 2D domains, while retaining the placement quality of Turk and Banks [98].

To study flow phenomena near wall regions or boundaries, i.e., curved surfaces in 3D, Rosanwo et al. [112] proposed a greedy seed placement algorithm. Similar to previous approaches, a single distance δ is used to control streamline density. However, the method avoids the computation of geodesic distances and reduces the search space for seed placement to a set of curves. The algorithm employs two sets of streamlines, namely, *primal* and *dual* streamlines. Primal streamlines are tangential to the vector field at every point and are used to visualize flow phenomena. Used to approximate the largest uncovered areas in the domain, dual streamlines are a supplementary set of streamlines that are orthogonal to the vector field at every point. A small set of both primal and dual streamlines can be initialized either randomly or by using flow field topology. Given the orthogonal directions of the two sets of streamlines, they intersect at

several locations. Segments of primal streamlines are stored in a priority queue P, ordered by arc length. Similarly, segments of dual streamlines are stored in a priority queue D, ordered by arc length. The algorithm iteratively selects the longest arc in P or D and places a seed at the midpoint to calculate the next streamline, followed by both queues being updated based on new intersections and segments. The algorithm stops when the length of the longest segment is less than twice the value of δ. An informed placement of the initial set of streamlines can reduce time to convergence for the algorithm and highlight flow topology resulting in speedups of 2x-3x and improved streamline placement quality over previous approaches [97, 39, 98] when evaluating streamline placement for planar surfaces.

Zhang et al. [113, 114] proposed a method to place streamlines in parallel for 2D flow fields. They define local tracing areas (LTAs) as subdomains enclosed by streamlines and/or field borders, i.e., regions where the tracing of streamlines is localized. Using an irregular domain decomposition strategy, the initial LTA is recursively partitioned into hierarchical LTAs. Within an LTA, if a valid seeding area (VSA, determined by streamline proximity criteria) exists, a new seed point is placed at the centroid of the biggest VSA. They use a cell marking technique, instead of performing distance checking, to mark zones where seeds can be placed and streamlines traced. The authors further extended the algorithm to support multiresolution and 3D flow fields [115, 116]. A comparison with Mebarki et al. [39] showed equivalent or better placement quality and an order of magnitude faster computation on parallel hardware.

Analysis: View-independent algorithms inherently provide coverage of the entire domain, i.e., object space, either by generating nearby candidate seeds from existing streamlines or by placing seeds in the largest voids in the domain.

However, given the primary objective is to achieve a desired density of streamlines, none of these algorithms have a focused ability to capture a ROI. Instead these algorithms primarily operate on the concept of maintaining a minimum separating distance. As a consequence, generated streamlines can be similar to existing streamlines since there is no measure to account for redundancy. Concerning computation, the majority of these algorithms adopt an iterative serial algorithm. Tracing streamlines one at a time, would limit applicability when considering large vector field data sets across multiple nodes or under in situ constraints.

View-independent algorithms support fast exploration once a set of streamlines is generated given only a single set of streamlines serves all viewpoints. Further, these techniques are typically limited to planar or curved surface flows, or are used in conjunction with other techniques such as texture-based flow visualization or interactive methods to address occlusion in volume flows.

9.4.1.2 View-Dependent or Image Space Techniques. By only considering object-space, view-independent techniques did not address occlusion problems that pose a significant challenge when exploring volume flows. View-dependent techniques presented in this section, take the image viewed by the user into account or use the current image as a guide to determine the placement of seed points and the selection of streamlines. The remainder of this section is divided based on whether algorithms use filters, image space seeding, or occlusion and projection of streamlines.

Algorithms Using Filters

Pioneering work in the field of streamline placement, Turk and Banks [98] proposed the use of a stochastic mechanism to iteratively refine the placement of streamlines to visualize 2D steady state flow. The approach is based on the idea

that for a given image containing a set of streamlines, the application of a low-pass filter to its corresponding binary image should result in an evenly-gray image if the streamlines are uniformly distributed. Areas with streamlines cluttered will have bright pixel values while sparsely represented areas will remain dark in the low-pass filtered image. The energy of the streamline image can be quantified as the sum of difference with a given gray-scale value at each pixel of the low-pass filtered image. The density of streamlines can be controlled by adjusting the size of low-pass filters and optimization of the streamline distribution is realized via iteratively minimizing the energy function. For this work, the filter applied is a circularly symmetric filter kernel from a basis function of cubic Hermite interpolation.

Beginning with streamlines generated from seed points at vertices of a 2D grid, where each streamline has an associated energy contribution, the set of streamlines is modified until the desired energy threshold is reached. The algorithm considers moving, lengthening, shortening, deleting, inserting, and combining streamlines based on energy. The streamlines modifications are either proposed by an oracle (50%) or are random (50%) to prevent any oracle bias. The oracle speeds up the convergence of the optimization by a factor of 3x-5x. To propose effective changes, the oracle uses image information to identify sparse regions and maintains a priority queue of streamlines based on energy level. Thus, the oracle suggests regions to insert streamlines or how to reduce the energy contribution of the most energetic streamlines. If the modification lowers the overall energy value of the image, the change is accepted, otherwise, the change is rejected. The process continues until the energy function reaches a threshold or the accepted changes are rare. Although this approach produced high quality streamline placement, it is computationally expensive given long convergence times.

Mao et al. [117] extended the Turk and Banks algorithm to uniformly distribute streamlines on a curvilinear grid. They use the image-guided algorithm because density distribution on curvilinear grids, which are anisometric, is hard to achieve with distance-based approaches. First, a mapping of vectors on the curvilinear surface to computational space is performed. To account for the mapping distortion caused by an uneven grid density on a curvilinear grid, a new energy function is employed. Using a Poisson ellipse sampling to distribute a set of rectangular windows in computational space, the streamline density is locally adapted to the inverse of the grid density in physical space. Use of such an energy function ensures the generated set of streamlines are evenly distributed after being mapped back onto the 3D surface.

Algorithms Using Image Space Seeding

Uniformly distributed streamlines in 3D space are not guaranteed to be evenly spaced in their 2D projection. To avoid clutter in a 3D streamline visualization, Li et al. [118] performed seed placement and streamline termination in image space, and streamline advection in object space. The algorithm operated similar to Jobard's and Lefer's algorithm [97], except that candidate seed points for a streamline were d_{sep} apart from the streamline in image space. Thus, even though a 3D volume flow is under consideration, for every sample point of the streamline only two possible candidate points are identified. A streamline is advected in object space and terminated if it is within d_{sep} from another streamline in image space. Further, a streamline closer to the viewpoint is preferred to another far behind. To support importance-driven seed placement, their algorithm decoupled seed point generation and streamline spacing control. A set of seed points is produced using a process that stochastically generates more seeds in a ROI, followed by tracing the

corresponding streamlines in object space. To avoid clutter, streamlines that violate spacing requirements in image space are deleted. This approach by Li et al. was the first work which used an image space based seeding strategy.

Spencer et al. [119] presented an evenly-spaced streamline seeding algorithm for vector fields defined on surfaces in 3D space. The algorithm is capable of generating both sparse and dense representations of the flow and can handle large, complex, unstructured grids with holes and discontinuities. Streamlines are only integrated for the portions of the surface visible in image space. The advection strategy removes the need to perform streamline tracing on a triangular mesh and instead projects the vector field onto the image plane. Seed placement and streamline integration are then performed in image space. The flow data is stored in a "velocity" image where each pixel stores the flow velocity on the surface and a 16-bit representation of the z-buffer representing the distance of the surface. The use of a z-buffer allows the algorithm to disregard non-visible portions of the surface and plays an important role in detecting discontinuities or edges. The algorithm places seed points, called grid-based seeds, in every cell of the mesh with non-zero depth. Next, it generates vector field-based seeds, i.e., candidate seed points, in a manner similar to Jobard's and Lefer's algorithm. They terminate a streamline when the proximity to another streamline drops below d_{test} or when z-buffer drops to zero or the change in z-buffer exceeds a user-defined threshold. Using both sets of seeds in combination ensures all visible sections (there are potential geometric discontinuities arising from edges and occluding surfaces) have a uniform distribution of streamlines. To avoid terminating streamlines near edges due to proximity in image space (greater distance apart in object space) they check for approximately the same z-buffer value. To improve depth perception in the

visualization, the value of d_{sep} varies with depth. The idea of reducing any complex surface to a 2D problem results in a computationally efficient algorithm. Spencer et al. used a GPU to improve rendering times and showed their streamline generation is faster than an implementation of Jobard's and Lefer's algorithm in 3D object space.

Algorithms Using Occlusion and Projection

Given the extensive use of contours to visualize scalar fields, Annen et al. [120] introduced the concept of vector field contours for flow exploration. The proposed algorithm generates isolated streamline which display behavior similar to that of isocontours. The approach is view-dependent in that seeding structures are identified by locating points where the dot product of the view direction and the vector field is zero, and a seed which takes one infinitesimal integration step preserves that condition. Multiple rendering passes are applied to extract the seeding structure with curvature being used in a similar manner as an isovalue in a scalar field. Streamlines are then integrated forward and backward until the dot product of the vector at the streamline position and the view direction exceeds a threshold. The extraction and rendering of the vector field contours is inter-frame coherent, with the flow field capable of being interactively explored.

Marchesin et al. [86] selected streamlines that contribute to understanding flow field characteristics, while simultaneously accounting for cluttering for a given view. The approach uses streamline features and the occlusion caused by it to decide whether to include a particular streamline. The four stage algorithm begins with the computation of a random pool of streamlines. Projecting all the computed streamlines onto an occupancy buffer helps identify highly occluded regions for a given view. Given the importance of swirling lines to understand

flow behavior, the occupancy buffer does not account for self-occlusion caused by a single streamline and simply measures the screen space footprint. Next, for each pixel, the number of streamlines projecting onto this pixel is calculated. The third stage, a pruning step, evaluates information conveyed and occlusion caused by a streamline. To determine the quantity of information conveyed by the streamline, the linear and angular entropy values of segments of a streamline, i.e., the local length and angular variation, are used. Additionally, they consider an overlap value to determine the occlusion caused for a given view. Combining these values, they present a streamline metric which is a weighted sum of the linear and angular entropies divided by the average overlap. Sorting streamlines by their score, the streamlines with the lowest score are iteratively removed, followed by an update of the occupancy buffer, and affected streamlines. The final stage of the algorithm decomposes the occupancy buffer into a number of tiles and computes the average occupancy for each tile. Seeding a small pool of random streamlines from the tile with the lowest occupancy, the streamline resulting in the least occlusion is retained. This process is repeated until all tiles have a non-zero occupancy. The approach captured features of the flow better than previous view-dependent methods and required a GPU for fast computation.

Günther et al. [87] presented an interactive, view-dependent, and inter-frame coherent flow visualization technique whose results are dependent on user-driven seed placement. The method has an initial preprocessing step, which involves both user-guided seed placement using a seed box, and random placement to generate streamlines that cover the entire flow field. For each streamline, a *screen contribution* value is computed by using a cubic Hermite interpolation function to map the number of visible pixels of the streamline to a transparency value. The

screen contribution values are used to determine which streamlines are visible to the user for a given view and fade out streamlines with only minor contributions. Given one important region of the flow can occlude another, the user can selectively place seed boxes in order to focus on certain regions. To support exploring regions of coherent flow, the user can highlight a set of similar streamlines by selecting a single streamline. Streamlines in a limited screen-space neighborhood window of the selected streamline are evaluated for similarity using linear and angular entropy.

Günther et al. [91] extended their previous work [87] by adopting a global line selection strategy. Starting with an initially dense domain sampling, the algorithm computes the opacity for every streamline segment in the field as a solution to a bounded-variable least-squares optimization problem. Metrics such as curvature, linear entropy, angular entropy, scalar entropy, segment length, or *screen contribution* can be used as an importance measure of a streamline segment. Depending on the metrics chosen the algorithm highlights relevant features in the flow field by minimizing the occlusion caused by other streamlines. While the optimization problem is based on the total number of streamline segments in the flow, the number of segments increases significantly and can become a bottleneck when considering unsteady state flow. To tackle the challenge of 3D unsteady flow, Günther et al. [121] modify their approach and employ a hierarchical representation of an integral curve and consider only a view-dependent set of candidate segments for the optimization process. Günther et al. use the GPU to achieve frame coherent, time coherent, and interactive flow exploration, thus improving on previous research.

Ma et al. [122] presented a view-dependent streamline selection algorithm that evaluates the information content of streamlines. As a preprocessing step, a

dense set of streamlines intersecting every voxel in the domain is computed. Next, for every sample viewpoint, the streamlines are sorted on the basis of importance. The streamline importance measure consists of entropy (considering both direction and magnitude) measured along the streamline, an evaluation of how much entropy is preserved for a given 2D projection, and a shape characteristic metric indicating whether the streamlines characteristics are being conveyed for a given viewpoint. The last two factors together form a view-dependent importance measure for each streamline for each viewpoint. First, a set of view-independent representative streamlines are identified by inserting streamlines into a priority queue based on the summation of their view-dependent importance measure for every view. A minimum threshold distance is used to avoid selecting redundant streamlines. To generate the view-dependent set of streamlines, the top-ranked streamlines for a viewpoint are combined with the highest rated streamlines from the view-independent set. Further, to maintain coherence as the viewpoint is changed, streamlines from a previous viewpoint are retained. A density map is employed to determine uncovered regions before rendering the final visualization. Their algorithm was able to generate fewer redundant streamlines compared to Marchesin et al. [86].

Analysis: View-dependent or image space algorithms have largely been proposed to account for occlusion that arises from visualizing streamlines in 3D. With respect to ROI, these techniques have evolved from initially only considering uniform placement on planar or curved surfaces, to limiting streamline calculation to the image space, to evaluating occlusion, projection, and information conveyed by a streamline before selection. Thus, the current state of the art includes view-dependent algorithms that are capable of highlighting ROI for a given viewing

angle. However, in general, these methods do not identify or strategically place seeds closer to ROI. Additionally, an important feature can occlude a second important feature resulting in only the streamlines in the foreground being selected. Most view-dependent algorithms do not consider similarity between streamlines selected and thus selections can be redundant if pruning steps are not performed. Early methods using filters were iterative and required long convergence times, whereas more recent algorithms are faster and provide interactive exploration of the flow field. However, during exploration, streamlines need to be reselected for every change in viewing angle and frame coherence techniques need to be adopted. View-dependent techniques would be suitable when considering automated in situ flow visualization or in an interactive setting with a user responsible for selecting the appropriate viewing angle.

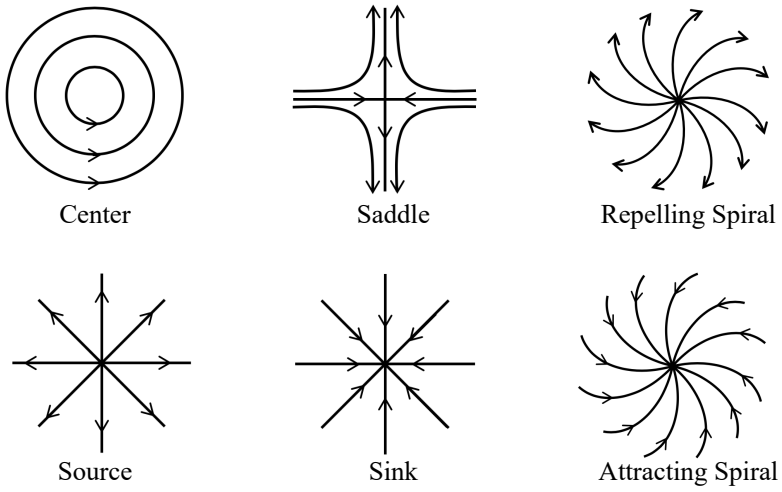


Figure 48. Types of critical points in 2D flows.

9.4.2 Feature-Based. Feature-based techniques make use of available vector field information to guide the seed placement or streamline selection. Prioritizing coverage of ROI of the flow field over a uniform distribution, these

approaches aim to first take measures to ensure seed placement occurs near salient flow features (e.g., critical points shown in Figure 48). We classify feature-based techniques depending on whether they explicitly extract flow field topology (9.4.2.1) for precise information or allow derived (9.4.2.2) and user-defined (9.4.2.3) scalar field to guide the algorithm. Approaches use a derived scalar field in order to either capture some particular flow field behavior or as an alternative approach to capture salient flow features without explicitly calculating their locations. In addition to strategies to highlight features, feature-based techniques often utilize density-based strategies to calculate streamlines in less interesting regions.

9.4.2.1 Flow Topology. Flow topology-based techniques calculate the locations of critical points or the separatrices and then use the information for seed placement. The remainder of this section is divided based on whether an algorithm uses critical point locations or directly uses separatrices as an initial set of representation streamlines.

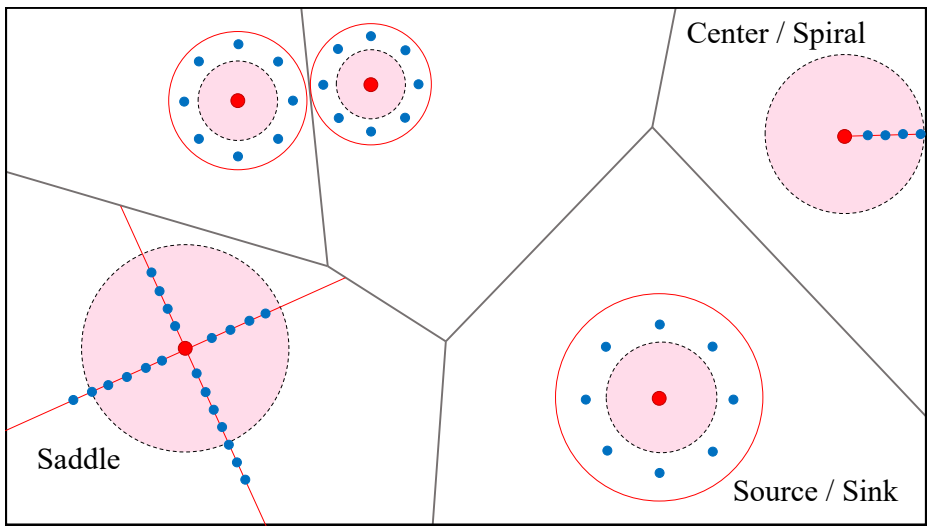


Figure 49. Images show seed templates for various critical points and regions of influence identified by the flow-guided algorithm. Example inspired by Verma et al. [15].

Algorithms Using Critical Point Locations

To visualize nested weather models, Treinish [123] proposed to use a combination of critical point analysis and a filter similar to Turk and Banks [98]. Deriving a set of seeds using a low-order approximate critical point analysis, an initial set of streamlines is computed. A low-bandpass filter is subsequently applied to the entire forecasted velocity field, as opposed to an image of the streamlines, to identify regions with a relatively large change in wind speeds. Seed points are placed in these regions to calculate additional streamlines. The technique was superior to using uniformly sampled seed points and captured detailed features from the forecast. However, this particular work did not provide seed placement specifics in relation to critical points or ROI.

Verma et al. [15] proposed the use of critical point-specific seeding template to capture flow behavior in the vicinity of critical points. The algorithm first identifies the locations and types of critical points in the 2D flow field and then segments the domain into approximate critical point neighborhoods. For the approximation, a Voronoi diagram partitions the flow into regions containing similar flow behavior. A second objective of the approach is to provide sufficient coverage of non-critical regions. After tracing long streamlines using the template seeds, a region of influence is determined for each critical point. In the spaces outside the regions of influence, Poisson disk distribution [124] is used to place additional seed points, with streamlines generation following Jobard's and Lefer's algorithm [97]. Figure 49 shows the seed placement templates, and a sample field with critical points, corresponding templates, and regions of influence, and the field partitioning such that each partition contains a single critical point. The algorithm was able to better capture flow behavior around critical points in both dense and

sparse flow representations when compared to previous techniques [98] while being computationally faster when selecting a greater number of streamlines.

In addition to extending the template-based approach to 3D steady state flow, Ye et al. [125] proposed improvements to the Verma et al. algorithm [15]. To account for the distance between and relative strength of critical points, they change the shape of the templates by mapping how eigenvalues of one critical point evolve into the eigenvalues of another. To determine the size of seed templates, separating regions are calculated. To calculate separating regions in 3D for each critical point, instead of using expensive full topological analyses [126, 127], an approximation is used. The size of the seeding template is set to a quarter of the distance to the nearest other critical point. Poisson sphere distribution is used to fill areas between critical point regions of influence. The algorithm involves a streamline filtering step to provide a less cluttered visualization. Streamlines are filtered on the basis of length, accumulated winding angle, and proximity to other streamlines. First removing short streamlines with low winding angles, followed by identifying a single representative streamline for a set of streamlines that have a similar start, end, and centroid location. Finally, for a cell with a high streamline count, denoting a dense region, streamlines with high winding angles are removed.

Liu et al. [128] proposed an evenly-spaced streamline algorithm (ADVESS) that employs two queues of candidate seeds. The primary queue consists of an initial set of seeds generated using the templates used by Verma et al. [15]. Candidate seeds generated from the initial location of the streamline seeds are also added to the primary queue to maximize the effect of the seeding patterns. The secondary queue, used only if the primary queue is empty, consists of candidate seeds generated along the calculated streamlines. As an optimization, cubic

Hermite polynomial interpolation using large sample spacing is used to represent a streamline using fewer points. As a consequence, the cost of distance checking is reduced due to less samples points considered. The algorithm terminates when all the seeds are processed, i.e., the queues are empty.

An additional improvement on previous evenly-spaced streamline algorithms is the use of an adaptive d_{test} value based on the local variance measured at each grid point in the 2D field. Appropriately scaling the value of d_{test} causes fewer cavities in the streamline placement. Further, the authors propose a robust loop detection technique which limits a streamline loop to a single cycle [129]. Employing the algorithm as one part of a hybrid seed placement approach, Liu et al. [130] presented a view-dependent approach for seed placement on a planar or curved surface. Using the double queue strategy differently, Poisson disk distribution is used to push a set of seeds to the secondary queue and begin the process. Candidate seeds introduced by the seed location of the accepted streamline are stored in the primary queue, and other candidate seeds are stored in the secondary queue. The approach is used for the purposes of image space seed placement to fill spaces after a primary set of physical space seeds are used to generate streamlines that are reused and lengthened between view frames. The combination of the two strategies provides a temporally coherent visualization. In comparison to previous approaches, the algorithm achieved placement quality better than Jobard and Lefer [97] and as good as Mebarki et al. [39] with loop detection, in addition to being computationally faster than both.

Ding et al. [131] present a technique to maintain temporal coherence when viewing unsteady 2D flow fields by using a moving mesh method. Another extension of Jobard's and Lefer's algorithm, they first extract critical points

to calculate an initial set of candidate seeds. Using Poisson disk distribution, they place seed points in ROI and add them to the queue of candidate seeds. They move the seeds towards the critical features by creating and deforming an auxiliary mesh along with the evolution of the vector field. They use a similar feed forward pipeline system to identify corresponding streamlines to maintain temporal coherence across frames.

Algorithms Using Separatrices As Initial Set

Chen et al. [132] modified Jobard's and Lefer's algorithm to highlight the vector field topology. Motivated by the visual discontinuity in periodic orbits and separatrices in current techniques, before using the seed placement algorithm, they first extract periodic orbits and separatrices and make them the initial streamlines. Further, to avoid clutter near sources, sinks, and periodic orbits, they terminate a separatrix if it is within a user-defined distance from the non-saddle end.

Preceding the parallel hierarchical LTA algorithm presented in Section 9.4.1.1, Zhang et al. [133] proposed to extract the flow topology and use the topological skeleton as the initial set of streamlines that segment the field. Next, additional streamlines are calculated by placing seed points at the center of topological areas in a recursive manner creating an approximately uniform distribution of streamlines. They extend the vector field domain in each direction by adding a layer of mirrored boundary cells [134]. Using the additional critical points from the extended vector field helps capture open separation and attachment lines.

Wu et al. [93] similarly extracted the flow field topology and partitioned it into regions of uniform flow behavior. However, as opposed to adopting a recursive method, they search for the longest path that orthogonally crosses all

streamlines within a region. Seeds are placed evenly along the longest path to produce approximately uniformly placed streamlines. They treat periodic orbits and saddle-connected loops as special cases. Using vector field reconstruction error as a quantitative measure for comparison, they demonstrate superior streamline placement than previous works [97, 39, 128, 132]. Their study shows that as a sparser set of streamlines is used, i.e., as the minimum separating distance increases, their algorithm results in lower reconstruction error in comparison to other algorithms.

Analysis: Flow topology-guided techniques have a primary objective of highlighting ROI in the flow field. With respect to redundancy, additional streamlines generated to fill empty spaces of the image or object space can introduce redundant streamlines given the use of only minimum separating distances. Computing the flow topology robustly for large scale complex vector fields is challenging given small changes in the flow field can lead to vastly different topological connections. However, for smaller scale problems, computing the flow topology is tractably. The majority of these techniques are most appropriate for a planar or curved surface flow given they either do not extend to 3D or result in occluding streamlines in 3D. For 2D domains, these methods have been demonstrated to have low reconstruction errors, which is highly desirable for multiple applications.

9.4.2.2 Derived Field. The topological structure of the flow field often shows a strong correlation to fields that can be derived from the vector field. Thus, several research efforts have leverage this property to propose the use of derived fields as an alternative approach to guide SPSS to capture salient flow

features. The remainder of this section is divided into algorithms that use entropy, derived vector field characteristics, or user-defined scalar fields.

Algorithms Using Entropy (Information Theory)

We can measure the amount of information or uncertainty in a local region using Shannon's entropy $H(X)$, where X is a random variable that models the input vector field.

$$H(X) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i) \tag{9.3}$$

A second key concept is conditional entropy. If Y models the visualization output consisting of streamlines, the conditional entropy between two variables $H(X|Y)$ informs how much uncertainty in the input data X remains after a set of streamlines in Y are displayed. This measure allows more streamlines to be placed in regions whose information is not sufficiently captured. The entropy of a vector field can be computed for each grid point in the domain and provides insight into the variation among vectors for a given neighborhood. Further, the entropy of a streamline is the accumulated value of entropy at sample points along the streamline, and the entropy of a voxel is the average of entropy values at the grid points of a single voxel.

Furuya et al. [135] considered streamline selection in combination with scalar field isosurfaces. The algorithm begins by integrating a large number of streamlines. The entropy of segments of individual streamlines is measured and used as a basis for selection. The measure of streamline entropy accounts for occlusion caused by isosurfaces by penalizing a streamline if segments of the curve are occluded. Finally, streamlines are sorted and selected in order of highest entropy. To control density, streamlines are only rendered if they are some minimum threshold distance away from existing streamlines.

Xu et al. [16] presented a framework that evaluates the effectiveness of a visualization by measuring how much information in the original data is being communicated. In this work, they empirically demonstrate that entropy in regions near critical points and separation lines is higher than that of other regions. Modeling a vector field as a distribution of directions, entropy is used to measure the information content in the vector field. The effectiveness of the streamline placement is measured by reconstructing a distribution of vectors derived from the selected streamlines. The approach begins by iteratively placing seed points in regions of high entropy. The authors use a diamond shape template to place 9 seed points in 2D and an octahedral shaped template consisting of 27 seed points in 3D. Further, to prevent large voids, seed are placed proportional to the computed conditional entropy. The streamline addition process ends when the value of conditional entropy of the entire domain converges to a small value. A final pruning step removes redundant streamlines. The algorithm provided quantitative control of the selection of streamlines.

Lee et al.[136] extended the framework presented by Xu et al. [16], to support a view-dependent streamline selection aimed at minimizing occlusion and revealing important flow features. Using the derived entropy field, a maximal entropy projection (MEP) frame buffer is computed for a given image space. The MEP buffer stores maximal entropy values, as well as the corresponding depth values for the given viewpoint. To identify the optimal viewpoints, i.e., views that convey maximum entropy information, MEPs of 780 viewports are evaluated. A streamline is assigned a higher priority if it reveals the flow near salient features and a lower priority if it occludes an important region of the flow. Streamlines are segmented and each fragment in the image plane is evaluated using information

stored in the MEP buffer, i.e., depth and entropy, to compute a scalar score ω. The streamlines are prioritized based on their value of ω which can be positive or negative. To maintain a streamline density proportional to the flow complexity, the screen space is divided into tiles and an expected streamline density equal to the average normalized entropy of the region in the MEP buffer is computed. For a given streamline, if the addition of the streamline affects more tiles with a density lower than the expected density, the streamline is added. This approach favors streamlines that reveal salient flow features without occluding other more important features. They demonstrated improved feature capturing compared to Marchesin et al. [86] for a view-dependent selection of streamlines. Further, athough they significantly benefitted from using a GPU, the serial selection is a bottleneck.

Ma et al. [88] present FlowTour, a framework that selects best viewpoints to explore a flow field visualized using streamlines. A skeleton-based seeding algorithm is employed to generate a set of streamlines that capture critical regions of the field. Variation of both direction and magnitude of vectors are considered to compute the entropy of voxels. Critical regions are identified as local neighborhoods in which voxel entropy values exceed a threshold. Sufficiently large regions of connected voxels with high entropy are used as input to a volume thinning algorithm that extracts the skeleton points. Skeleton points are connected by applying a minimum spanning tree algorithm to produce a tree-structured skeleton line. The density of streamlines is controlled by evenly-spaced seed placement along the skeleton. Candidate viewpoints are generated on the basis of the critical regions identified in the field. Finally, best viewpoints for each region are selected and connected into a view path using a B-spline curve.

Algorithms Using Derived Vector Field Characteristics

The problem of image space cluttering is exacerbated in 3D unsteady state flow, given pathlines can intersect in space. To study unsteady state 3D flow, Wiebel et al. [137] introduced the concept of an eyelet. Calculating a set of pathlines that pass through the same single point (eyelet) in space at different times yields an insightful static visualization of the unsteady flow field. The collection of pathlines can be used to construct a surface to visualize the flow. If pathlines diverge more than a user-defined threshold, a new seed is added at the eyelet at a time step in between the time steps of its neighboring particles. While this approach is conceptually similar to Hultquist [95], instead of adding a new point at the location where divergence is detected, it is added at the eyelet to guarantee the particle would indeed pass through the eyelet. The placement of the eyelet plays a critical role in the usefulness of the algorithm to study the flow field. They identify regions of high activity by introducing measures to capture the change of a vector field over time. A dot product variation is computed by accumulating the positive dot products of vectors in consecutive time steps. A second measure, the vector variation is the norm of the computed difference of the two vectors considered. Isosurfaces drawn using these variation fields help identify regions of high and low activity for further investigation. Additionally, edges and corners, or regions behind flow passed objects, singularities, and vortex cores serve as good locations to place an eyelet.

Wang et al. [138] visualized explosion fields by first generating an isosurface of the magnitude of velocity. The isosurface region is then divided into a series of subregions that are almost equal in area. The center of each subregion is used as

a seed point with the integrated streamlines always starting from the center of the explosion and extending outward in a direction perpendicular to the isosurface.

Luo et al. [139] proposed a technique that combined the use of derived scalar fields and topological methods, such as contour trees and persistent homology. Global importance is measured in terms of persistence of topological features in the vector field and streamline density in the generated visualization is used to reflect the same. Hodge decomposition [140] is a technique used to convert a 2D vector field into two scalar fields gradient potential and curl potential. Maxima and minima of the gradient field correspond to sources and sinks respectively and guide the placement of a set of *gradient* seeds. The number of seeds placed is proportional to the persistence of the maxima and minima, measured by determining the amount of perturbation required to smooth out the mountain peak or valley. A contour tree encodes the evolution of level sets of the curl field and is used to generate a set of *curl* seeds. Each branch of the contour tree corresponds to a topological component of the domain. Each branch is assigned a number of seeds proportional to its range function to collectively produce a set of curl seeds. Every seed location is evaluated to determine gradient vector or curl vector magnitude dominance at that position. Only gradient seeds in positions of gradient dominance are used. Similar, only curl seeds in positions of curl dominance are used. Luo et al. demonstrated superior placement quality in terms of reconstruction error compared to Li et al. [90] and Xu et al. [16].

Yu et al. [17] used the curvature and torsion of the vector field to generate a saliency map to guide seed placement. The saliency map is computed as the difference between Gaussian-weighted averages of curvature and torsion fields calculated at multiple scales, i.e., varying the standard deviation of the

Gaussian filter. The saliency map is computed for five threshold distances and then all five saliency maps are combined with a nonlinear normalization. While the computation of the saliency map is relatively expensive on CPUs, it can be computed within a few seconds on a GPU. Given a final saliency map, the seed placement algorithm selects locations in order of decreasing saliency. Long streamlines are favored, with streamlines integrated until they reach a critical point or leave the domain. To reduce redundancy, streamlines that occupy the same voxel as an existing streamline are discarded. The use of the saliency map as opposed to directly seeding based on the curvature or torsion fields, allows streamlines to be placed closer to critical points. Further, the generated set of streamlines is hierarchically clustered to enable exploration at different levels of detail and manage clutter (details in Section 9.4.3.1).

Zhang et al. [141] investigated the usage of a scalar field Φ derived from the input vector field by integrating the rotation of the integral curves. Seeds are placed where $|\nabla\Phi|$, the magnitude of the rate of variation of the derived field Φ, is greater than a user-defined threshold. Randomly starting from a placed seed point, an integral curve is computed, followed by the filtering of nearby seeds. The process is repeated with the remaining seeds.

To visualize a vortex rope that builds up in the draft tube of a water turbine, Bauer et al. [142] proposed a particle seeding scheme to visualize unsteady flow. They use Sobol quasirandom sequences [143] to obtain a uniform distribution while avoiding clustering and artifacts like regular patterns. Given the vortex rope is a rotating helical structure, the helicity in the field is evaluated to identify ROI. New particles are introduced to regions with a scalar value of helicity greater than a predefined threshold by offsetting the original point set of quasirandom sequences.

They use a layer of invisible buffer cells that enable particles to fade in and out smoothly from the ROI. Guthe et al. [144] presented another seed placement approach aimed at distributing more particles in ROI. The seed placement is based on an adaptive sampling of the field with the goal of achieving a higher sampling resolution in more interesting regions and a lower sampling resolution in less interesting regions. The local gradient, divergence, and curvature of the vector field is used to influence the particle distribution. Additionally, local shear and rotation of the vector field or distance to the closest critical point can be used. An octree data structure is employed to maintain the distribution of particles in the domain. The distribution octree is updated as particles travel along streamlines with particle age being a deciding factor in regard to particle removal in overcrowded regions.

Particle-based visualization systems have seen efforts to improve the interactivity of flow visualization [145, 146, 147, 148]. Engelke et al. [149] proposed a particle system that results in an adaptive particle density by using autonomous particles. Particles operate in parallel without neighborhood information or inter-particle communication by following a set of rules. The rules dictate particle birth, death, and split events that influence the density of particles in different regions of the flow. The study uses split criteria such as λ_2 [150], the curvature of the particle trajectory, and distance to an object in the field. The parallel nature of the system allows interactive visualization while maintaining a smart sampling of the flow. Both context and feature particles are used, with context particles being randomly introduced into the domain to prevent underrepresented regions. Feature particles are children of context particles and are introduced when a split event occurs. Split events are determined by a combination of properties such as energy, generation of the parent particle, and the local importance measures in the flow.

Analysis: Several studies demonstrated the use of derived fields such as entropy, curvature, and torsion to highlight salient features of a flow field using streamlines. Additionally, derived fields provided a means to visualize regions of maximum interest irrespective of the depth from a given viewing angle. To reduce redundancy the use of a pruning step to remove redundant streamlines was demonstrated, however, minimal redundancy is not the focus of these algorithms. Deriving fields from the vector field can often be performed in parallel and computed relatively fast without considerable scalability issues. Thus, these techniques have the benefit of fast computation and the ability to highlight ROI. Considering these benefits, derived field techniques have the potential to be applied to particle-based flow visualization systems or within in situ flow visualization contexts.

9.4.2.3 User-Defined Scalar Field. The algorithms in this section are designed to use either a specific user-defined scalar field or are adaptable to utilize any given scalar field to generate a representative set of streamlines.

Zockler et al. [151] proposed to use a statistical method to facilitate placement of streamlines with a density proportional to some scalar quantity. Considering a uniform grid over the domain, for each cell, a local degree of interest is computed. Cells are selected on the basis of the parameterization of a probability distribution using the local degree of interest. Seeds are then placed in those cells, with streamlines grown for a fixed length with forward and backward integration. For scalars ranging over multiple orders of magnitude, streamline distribution can be unsatisfactory. To address this problem, a histogram equalization approach is used to obtain a homogenous distribution. Weinkauf et al. [152, 153] applied this technique using fields of curvature and torsion.

Schlemmer et al. [154] presented a heterogeneous distribution of streamlines based on a density map derived using scalar fields (temperature), derived vector field information (magnitude of velocity, vorticity), or a user-defined density function. Here, streamline density is the number of occupied cells over the total number of cells in a domain. To calculate *priority* streamlines, they first define a density map used to guide seed placement, with the map updated after every streamline calculation. The first seed is placed at the location of the maximum value of the initial density map. The next location is chosen as the furthest of the next five maximum values. Given the density map is monotonically decreasing over time as streamlines are added, the algorithm will eventually terminate.

Shen et al. [155] proposed the use of fractal dimensions [156] for streamline selection. Measured using the box-counting ratio [156], fractal dimensions can provide insight into the complexity of a streamline by considering its space-filling properties [157]. The box counting ratio is measured by counting the number of cells a streamline intersects in a small grid. For each voxel in the domain, a scalar value is calculated using the local box counting ratio of streamlines that intersect the voxel. The scalar grid is used to filter streamlines by fractal dimension and to identify regions containing vortices and turbulence. Although the space-filling properties of streamlines are evaluated and desired regions are highlighted, it does not guarantee to capture ROI in the flow. The algorithm demonstrates ability to capture a feature focused set of streamlines and remove redundant curves.

Given a pattern template as input, Bujack et al. [158, 159] derived a scalar field that shows how similar each location in the vector field is to the template using rotation invariant pattern detection. Next, they seed streamlines with a

probability that is proportional to the calculated scalar field. This method is able to explicitly visualize ROI as defined by the user and minimize their occlusion.
Analysis: Algorithms using alternative scalar fields defined by the user are capable of highlighting ROI to the user. However, these are less traditional approaches to visualizing flow features and require users to define parameters. Given the diversity in the underlying approach, these methods demonstrated varying degrees of control with regard to minimizing redundancy. Pruning of streamlines can be easily integrated in a technique by controlling the number of streamlines passing through any cell. Lastly, these techniques can be relatively fast given the easy generation and use of a guiding scalar field in conjunction with straightforward algorithms.

9.4.3 Similarity-Based. SPSS techniques based on density distribution or feature extraction have certain drawbacks, i.e., redundancy, require feature extraction or derivations of guiding fields. Similarity-based approaches have been gaining popularity in the past decade due to their ability to overcome these drawbacks and by serving multiple flow exploration tasks. These techniques are based on the concept of first identifying similar streamlines and then selecting representatives from groups of similar streamlines. Additionally, these approaches support hierarchical grouping of streamlines for a level-of-detail approach and the identification of streamlines similar to a query streamline, i.e., isolating streamlines that match a given description. Similarity between streamlines is measured using spatial proximity (9.4.3.1), shape (9.4.3.2), or by employing machine learning (9.4.3.3) to cluster streamlines based on several feature attributes.

Streamline clustering and selection research has extended beyond the tasks mentioned above to include feature-specific selection, application to non-dynamic

vector fields or medical applications, and methods to improve costs of clustering. For example, in an effort to highlight structures of interest in the flow, Salzbrunn et al. [160, 161] define streamline and pathline predicates to cluster similar integral curves that satisfy some criteria. Clustering of curves is commonly used to visualize diffusion tensor imaging (DTI) data [162, 163, 164, 165, 166]. With respect to medical visualization, Oeltze et al. [167] evaluated three clustering techniques — k-means, agglomerative hierarchical clustering (AHC), and spectral clustering to reduce clutter when visualizing streamlines traced from simulated blood flow. Recently, Shi et al. [168] conducted an in-depth comparative study of several curve clustering and simplifications algorithms used for flow visualization to provide users with a systematic guideline to choose a specific approach. Lastly, given the high cost of similarity metrics that involve pairwise streamline comparison, Shi et al. [169] proposed metrics that run in linear complexity.

9.4.3.1 Spatial Distance. Streamlines in the proximity of one another are likely to have sections that display similar curves. One way to identify such streamlines is through spatial distance. The remainder of this section is divided into algorithms that use proximity as a measure of similarity or methods that use the mean of closest point distances as a similarity metric.

Algorithms Using Dissimilarity Metrics

Motivated by the shortcomings of density and feature guided methods, Chen et al. [92] presented the first similarity guided streamline placement algorithm for 2D and 3D steady flows. The algorithm naturally accentuates regions of geometric interest while minimizing streamlines in areas of parallel flow. As a measure of similarity between two streamlines, a similarity distance metric that has two influencing factors is defined. The first factor is a translational distance measured

as the Euclidean minimum distance between points on two streamlines. The second factor is a measure of shape and orientation similarity, and is measured over a spatial window. A spatial window is formed by identifying a predefined number of equally spaced sample points along the curve. Figure 50 shows sample point pairs for two streamline windows. The translational distance is the average

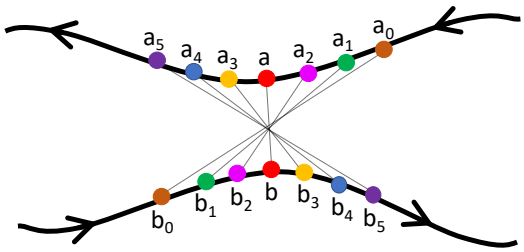


Figure 50. The figure shows the pairs of points in a spatial window used by Chen et al. [92]. The pairing considers velocity direction along the streamline and thus measures shape and orientation similarity.

deviation of sample point pair distances from the the center point pair distance. Starting from a dense set of candidate seed points, a streamline is traced until its similarity distance for a window falls below a pre-specified similarity tolerance. If the streamline length is greater than a minimum length threshold, the streamline is added to the set of selected streamlines. The use of shape and orientation for similarity resulsts in 30% better placement compared to only using the translational distance. However, the method is sensitive to the order in which candidate seed points were tested.

Li et al. [90] proposed an iterative algorithm to select a small set of streamlines in 2D steady state flow fields. The algorithm exploits the spatial coherence in a flow field to achieve a minimal selection of representative streamlines. The density of selected streamlines, each of which is integrated for

as long as possible, varies to reflect the different degrees of coherence in the field. Their algorithm derives local and global metrics by employing 2D distance fields that measure the distances from each grid point to nearby streamlines. The local metric measures the direction difference between the vectors of the original field and an approximate field computed from streamlines in the vicinity. The global metric measures streamline dissimilarity by accumulating the local dissimilarity at every integrated point along a streamline trajectory. To place seed points, the algorithm begins by placing a random or central seed point. Next, the streamline is integrated and local dissimilarity is evaluated at each grid point. A streamline is accepted if the local dissimilarity value of the original seed point is greater than a threshold and if the global dissimilarity value of the streamline is greater than a second threshold. The next candidate seed is picked by sorting the grid points into a sorted queue in descending order of the local dissimilarity value. The process ends when no remaining candidate seeds satisfy the dissimilarity threshold requirements. To optimize the algorithm, the number of candidate seeds is reduced by eliminating grid points on boundaries and by marking cells visited by rejected streamlines, i.e., streamlines with global dissimilarity values below the threshold.

Algorithms Using Mean of Closest Point Distances

The mean of closest point distances (MCPD) [163] is the mean of Euclidean distances between pairs of points formed by mapping each point of one streamline to the closest point of the other. MCPD has proven useful for multiple SPSS algorithms.

Building on the feature highlighting streamline generation technique presented in Section 9.4.2.2, Yu et al. [17] enable exploration at varying levels of detail via a hierarchical streamline bundling visualization. To calculate the bundles

of streamlines, i.e., clusters, MCPD is used as a similarity measure. Beginning with each streamline in a distinct cluster, they successively merge the two most similar streamlines in a bottom-up fashion until a stopping criterion is reached. Clusters are merged using the *single-link* method where the distance between two clusters is the minimum of the distances between all pairs of member streamlines. To represent each cluster of streamlines, as opposed to streamlines close to the cluster centroid, the union of streamlines along the cluster boundary is used. In addition to capturing sources and sinks, only boundary streamlines of a cluster best reveal the saddle together with other boundary streamlines of neighboring clusters.

While Yu et al. [17] used MCPD as a measure to merge streamlines into a cluster, Tao et al. [170] use the measure to identify redundancy and limit the number of representative streamlines selected. Measured using a *streamline information* metric, the selection is performed by considering the contribution of a streamline to a large set of sample viewpoints. Starting with a random pool of streamlines, a matrix containing the probabilities of seeing each streamline from each viewpoint considered is created. The probability of seeing a streamline is high if it contains a high amount of information in 3D and the 2D projection for a given view preserves the information well. Additionally, they score the shape characteristics of a streamline projection by evaluating each segment of a streamline subsampling and score segments higher if they form a 45 or 135-degree angle to the viewing direction. *Streamline information* represents the degree of dependence between a streamline and the set of viewpoints. A low value indicates a streamline contributes in a balanced manner to a large number of viewpoints, while a high value would indicate a streamline visible in a small set of viewpoints. Streamlines are sorted into a priority queue in decreasing order of accumulated streamline

information. However, these streamlines have significant redundancy and are likely cause clutter in 3D. Pairwise similarity between streamlines is measured using MCPD to avoid repetitive streamlines. Further, a *viewpoint information* measure is defined to similarly guide viewpoint selection for the chosen streamlines. In comparison to other works, the vector field reconstruction error for this algorithm is lower than both Xu et al. [16] and Marchesin et al. [86]. Han et al. [171] employ this technique in situ to save a compressed representative set of streamlines for post hoc flow analysis.

Opacity adaption is a technique used to manage occlusion and cluttering of streamlines in a 3D field. However, it can result in the loss of spatial perception when streamlines are faded out to reveal ROI. To retain the perception of spatial relationships between streamlines, Kanzler et al. [89] select a set of streamlines such that the screen-space density of the streamlines is locally adapted to the importance of the streamlines. The algorithm requires computing a fully balanced line hierarchy to facilitate the uniform removal of streamlines in the domain and obtain the desired density at run time. MCPD is computed for all pairs of streamlines and is used to define a fully connected distance graph. A minimum cost perfect matching algorithm is recursively used to identify pairs by minimizing the sum of all included edge weights, i.e., the similarity measure. Single linkage is used to merge clusters since it results in spatially coherent merging of clusters. Streamlines are selected by using visibility values based on (1) an importance measure, such as curvature, measured along the streamline, and (2) the occlusion caused by the streamline to other potentially more important lines. Visibility thresholds are assigned to lines in the hierarchy based on the level at which the line is the representative for its cluster. Further, the visibility values are then used

to locally control the line density. Although this method improves the spatial perception of the visualization, it incurs a long preprocessing time to build a balanced line hierarchy.

Analysis: Algorithms using spatial distance to measure similarity between streamlines were capable of selecting distinct streamlines that accurately captured flow features and provided a parameter to control redundancy. Calculating the similarity between large numbers of streamlines can be computationally expensive and require large preprocessing times depending on the approach adopted. The shortcoming of using a spatial distance for similarity measure is that the metric is limited to similarity between streamlines in proximity, i.e., similar streamlines present in different regions of the flow would not appear similar. This limits the use of spatial distance to the application of generating a representative set of streamlines.

9.4.3.2 Shape. Feature attributes have been extensively used to evaluate the similarity between streamlines. A common use case is to identify all streamlines similar to a given streamline. Distance-based similarity measures primarily account for proximity and are sensitive to rotation, translation, and scaling. Feature attributes address these shortcomings by evaluating similarity in a proximity, size, and orientation insensitive manner. The remainder of this section is divided into algorithms that use sample points or segments of a streamline to measure shape similarity between streamlines.

Algorithms Using Point Sampled Features

Wei et al. [172] proposed a technique to select streamlines similar to a user-sketched streamline. The user sketch is a 3D curve whose 2D projection is used as the input to the algorithm. The algorithm approximates the sketched curve

and the streamlines using an arc length parameterized cubic B-spline and samples the curvature at equal arc length intervals along the curve. Using a feature vector constructed by concatenating the curvature and torsion at sampled points, they employ a *string matching* approach to find similar streamlines. The difference between two vectors is measured using the *edit distance* [173]. The most similar 3D streamline is identified and used it as a reference for clustering. All streamlines similar to the reference streamline are selected as the result of the streamline query. Additionally, Wei et al. proposed to choose cluster representatives from an AHC scheme on the basis of view-dependent quality. Viewpoint quality of a streamline is computed by accumulating the winding angle of the 2D projection of the streamline, with larger values representing more information.

Zheng et al. [174] presented a streamline selection algorithm for 2D flow fields that uses streamline feature classification, similarity and entropy. Streamlines are first classified on the basis of the feature they highlight, i.e., a vortex, source, sink, or saddle, and are also prioritized in that order when they capture more than one feature. In addition to feature type, each streamline has a feature position, i.e., the vortex center for a vortex streamline, the critical point for a source-sink streamline, or the point of highest entropy for a saddle streamline. Next, streamlines are iteratively clustered on the basis of a combination of feature type and proximity of the feature position. A similarity metric that uses a combination of both geometric shape properties and proximity is defined. Curvature and accumulated angle are used as geometric shape properties. Sample points between two streamlines are mapped using Dynamic Time Warping (DTW) [175], which is a dynamic programming algorithm to find an optimal mapping between two sequences. For a proximity evaluation, MCPD is employed. The algorithm selects

a set of streamlines by identifying a streamline from each feature subset with the highest entropy accumulated along the points of the streamline. Streamlines within each feature subset that are least similar to the previously selected streamlines are picked next. The last step involves, selecting more streamlines by considering similarity to previously chosen streamlines and streamline entropy until a desired number of streamlines is selected. The algorithm limits the number of redundant streamlines and is capable of generating a placement qualitatively equivalent to the work by Yu et al. [17] for a 2D flow.

Algorithms Using Segmentation

To tackle the high computational expense of distance-based similarity measures which involve performing large numbers of Euclidean distance tests, McLoughlin et al. [176] propose to measure streamline similarity by first computing an integral curve-specific signature. The signature is computed by segmenting an integral curve and using a set of curve-based attributes, namely, curvature, torsion, and tortuosity, to describe the integral curve per unit length of the curve. The χ^2 test [177] is used to measure similarity between streamlines and is performed for all streamline pairs generating a similarity matrix that enables fast lookup for the clustering process. Additionally, given the streamline signature is proximity independent, a Euclidean distance measure can be used to supplement the χ^2 similarity measure. To address segment alignment issues when comparing a pair of streamlines, seed placement is limited to rakes orthogonal to the local flow and a hierarchical signature for each streamline is considered. The algorithm was demonstrated to be orders-of-magnitude faster than the approach by Chen et al. [92].

Chen et al. [178] use an entropy-guided seed placement strategy to generate an initial set of streamlines. Streamlines are clustered using a two-stage k-means algorithm. The first stage only considers the start, middle, and end point of a streamline for clustering. Each cluster after the first stage is further subdivided into clusters by considering the linear and angular entropy of streamline segments. The two stage k-means algorithm is chosen over single-linkage clustering with MCPD due to the quadratic computational complexity of the latter.

Lu et al. [179] proposed a similarity measure based on the statistical distribution of measurements along a streamline. As a result of being based on distributions, the similarity measure is less sensitive to length, spatial location, and orientation. First, streamlines are recursively segmented until a minimum length threshold is reached or a segment cannot be split into two segments which are dissimilar enough. Next, a 1D histogram is constructed to represent every segment, and a 2D histogram to represent the entire curve. The use of the 2D histogram is to capture the order of the segments, and thus avoid dissimilar streamlines with similar feature distributions appearing similar. Given streamlines may be represented by a varying number of segments, a mapping between two sets of segments is performed by using DTW. The difference between two histograms is measured using earth mover's distance (EMD). Curvature, torsion, and curl are used as measures along a streamline to demonstrate the distribution-based approach. Their proposed AHC scheme, in addition to using a distance measure (for example, *single-linkage*), uses a balance parameter that accounts for number of streamlines in a cluster and can force smaller clusters to merge early in the process to produce a more balanced tree. The algorithm was demonstrated to be much faster than using distance measures to identify similarity.

Li et al. [180] proposed to use a similar feature vector description approach to measure streamline similarity. To address the issue of dissimilar streamlines having very similar feature distributions, they use a feature descriptor that encodes the spatial relations among the features. The encoding mitigates the need to segment the streamline and find mappings between two streamlines during similarity evaluation. Beside using local streamline metrics like curvature and torsion, Li et al. use global geometric properties of tortuosity and velocity direction entropy to describe a streamline. A weighted Manhattan distance between constructed feature vectors of two streamlines measures the similarity between them. Following a pairwise similarity evaluation between all streamlines, *affinity propagation* [181] is used to cluster streamlines and form a hierarchy. The affinity propagation algorithm accepts the measured similarity values as input and simultaneously considers all the data points as possible cluster centers. It uses similarity values as preference values for each data point. The algorithm then exchanges real-valued messages between data points until it converges to produce a set of cluster centers of high quality.

FlowString is a framework for partial streamline matching proposed by Tao et al. [182] that models streamlines as strings. Streamlines are first resampled on the basis of winding angle, using a threshold small enough to capture relatively simple patterns of the streamline segment between neighboring sample points. All sample points are then evaluated pairwise for a similarity measure, the *Procrustes distance* (a metric to quantify similarity for 3D shapes and is extensively used in biological morphometrics), followed by applying *affinity propagation* for clustering using a GPU. The resultant clusters after two levels of affinity propagation serve as the local shapes for the data set. The local shapes are used as characters, which

together form an alphabet, using which words can be formed by concatenating characters together. Their work differentiates between approximate and exact searches; they achieve the former with dynamic programming and the latter with a suffix tree. This work was the first to pursue labeling and classification of streamline segments.

Analysis: Use of features along a streamline allowed for comparison between streamlines in a proximity, scale, and orientation invariant manner. Table 14 shows similarity measures and corresponding clustering techniques used. Streamlines were either identified by the feature attributes of points along the curve or curve segments. Thus, with respect to ability to identify ROI, these algorithms are capable of responding to streamline similarity queries in addition to calculating a representative set. In general, similarity-based methods provide the user with control of redundancy by allowing the number of clusters or similarity threshold to be varied. However, computing similarity between streamlines requires a one-to-one comparison and significant processing times. To accelerate the process, there are two major techniques: (1) using accelerators for clustering, and (2) use of segmentation to avoid large numbers of Euclidean distance checks. However, these similarity measures are being performed on a relatively small number of curves and extending these techniques to scale by using more computationally efficient measures is a current research area.

9.4.3.3 Machine Learning.

The application of machine learning techniques to scientific visualization problems has been a recent development in the field. With respect to flow visualization and specifically the use of streamlines, there has been recent activity.

Algorithms Using SVM For Segmentation

Reference	Similarity Measure	Clustering
[178]	Spatial, Shape Properties	K-means
[176]	χ^2	AHC
[179]	EMD	AHC
[180]	Manhattan Distance	Affinity Propagation
[182]	Procrustes Distance	Affinity Propagation

Table 14. Similarity measures and the corresponding clustering methods used by similarity-based techniques that use segmentation.

Streamline segmentation has growing importance given its application in identifying the similarity between streamlines. However, current measures of segmentation and similarity measurement do not account for human perception or what a human considers important. Li et al. [183] adopted a user-guided approach that used a binary support vector machine (SVM) to perform streamline segmentation. The approach begins by first generating a pool of random streamlines, followed by the use of affinity propagation for clustering based on a similarity metric that uses curvature and torsion 1D histograms. 1D histograms, formed by concatenating previously computed histograms of curvature (20 bins) and torsion (40 bins), describe the shape characteristics of a streamline. Next, users choose segmentation points along the set of representative streamlines from each cluster. For every segmentation point, the algorithm computes a feature vector comprising velocity direction ratio, tortuosity ratio, the curvature and torsion histograms, and the volume ratio of minimum-bounding ellipsoids using varying neighborhood sizes. User-selected segmentation points are positive training samples, while all non-segmentation points are negative training samples used to train an SVM classifier. The streamline segmentation process is carried out for the remaining streamlines using the classifier. If a group of nearby points is selected as segmentation candidates, a post-processing step chooses the point

with the smallest ratio of minimum-bounding ellipsoids as the final segmentation point. The algorithm presented by Li et al. is the first to use supervised machine learning for streamline segmentation. They demonstrated superior segmentation and feature capturing in comparison to previous similarity-based methods that used segmentation.

Algorithms Using DNN For Feature Description

Han et al. [94] used a DNN, named FlowNet, to identify a representative set of streamlines for a given flow field. An autoencoder, which features both convolutional and fully-connected layers, enables the network to learn a complex data representation by using both local and non-linear combinations of neurons. FlowNet accepts voxelized and downsampled representations of streamlines as input to learn features by non-linearly mapping each representation to a feature descriptor of 1024 dimensions. A binary cross-entropy loss function is employed to train FlowNet. To explore the feature descriptors generated, t-SNE [184] is applied for dimensionality reduction, followed by interactive parameter tuning of the clustering method DBSCAN [185] to find a suitable number of clusters or set the minimum number of samples in a cluster. A representative streamline is then identified as one who minimizes the sum of Euclidean distance to all other points in the cluster. Han et al. don't explicitly use any streamline attributes, but instead, are the first to employ a DNN to calculate flow features before clustering and selection. They demonstrated streamline selection which results in lower vector field reconstruction error (Table 15) compared to Tao et al. [170] and Xu et al. [16]. However, training the network can require days to complete.

Analysis: The use of machine learning for scientific visualization has increased recently. Machine learning has been leveraged to accurately capture ROI in a

Dataset	[94]	[170]	[16]
crayfish	**0.102**	0.116	0.144
solar plume	0.283	**0.280**	0.303
five critical pts	**0.023**	0.026	0.031
tornado	**0.080**	0.105	0.101
two swirls	**0.065**	0.070	0.071

Table 15. Vector field reconstruction error measures from the study by Han et al. [94]. Error is measured as the average angle difference between the original vector field and the vector field reconstructed using the streamlines.

volume flow and to perform streamline segmentation based on a user specification of what is considered interesting. The method by Han et al. [94] selects only representatives of clusters and thus minimizes redundancy. Currently, the two major drawbacks of these methods are (1) the need to subsample the data set in order for it to be feasible to process and (2) long run times. Further, the features learned are data set-specific and require each data set to be processed. Future efforts can aim to build a large database to train neural networks to identify flow features. Thus, there is potential for further research concerning the application of machine learning techniques to unsteady flow visualization and overall computational improvements.

9.5 Manual Techniques

Manual placement of initial seed positions is a common first step when using streamlines to study a flow field. Interactive flow visualization techniques were introduced two decades ago and have since evolved to give the user varying degrees of control of the generated visualization. While several interactive flow visualization techniques exist, in this section, we limit our study to manual techniques involving tools for placement of seeds, density control, and the use of domain information to do the same.

9.5.1 Use of Interactive Tools. The virtual windtunnel project [186] was an immersive virtual reality-based system used for the investigation of airflow around a Space Shuttle. In this work, Bryson et al. describe the use of a hand position sensitive glove controller for injecting particles into a 3D unsteady flow. To study ROI in the flow, such as boundary layers and turbulent regions, the locations of seed points are interactively selected and the corresponding pathlines are integrated. The resultant graphics objects can be visualized and manipulated. In addition to rapid seed placement, the environment supports repositioning, grouping of seeds as a rake, and deletion of existing seed points using hand gestures. Hardware limitations at the time (1998) meant spatial subsampling of the vector field was required for interactivity or use of a supercomputer with dedicated graphics resources.

Schulz et al. [187] aimed to improve the interactivity of a virtual reality-based exploration system. The target application, a car body development aerodynamics simulation, required particle tracing to account for collisions with the car body. The initial positions of seeds are specified using a freely movable probe similar to Bryson et al. [186] and aligned on a rake or inside a cube. Additionally, they proposed application-specific data structures and interpolation techniques for fast particle tracing.

Laramee et al. [188] proposed the use of a manual seeding tool which allowed six degrees of freedom. The seed placement tool is a two dimensional seeding plane grid. Initial seed locations are set at the grid points before streamlines are calculated. Besides offering grid resolution control, the seeding plane can be translated, rotated, and scaled enabling convenient seed placement options. Laramee [189] proposed another system, named Streamrunner, which

attempted to address problems of occlusion, and the lack of directional and depth cues when interactively using streamlines in 3D flow. Streamrunner gave the user control over seed placement and the evolution of streamlines from the time they are seeds until they reach full length. This provided users with a sense of direction of flow and depth when observing the growth of the streamlines.

9.5.2 Use of Domain Information for Direct Seed Placement.

To generate informative visualizations, the manual placement of seed points or rakes is most suitable when aspects of the flow field behavior are known and scientists can intelligently place seed points. Alternatively, seed points may be manually placed near local maxima of an interesting scalar derived from the vector field or near critical points of the vector field topology [194]. Several flow visualization works adopt this approach.

For certain applications, knowledge of moving objects in the domain or specific component design assists in deciding seed point locations. Engineers are often required to evaluate the pattern of flow, such as a swirl or tumble flow, in the combustion chamber of an automobile in order to achieve efficient and stable combustion. Laramee et al. [188] strategically placed a seeding plane near the intake ports of a combustion chamber from where fluid enters to evaluate swirl flow. Multiple seeding planes are manually placed and the length of the generated streamlines in the combustion chamber to capture tumble motion is limited. In another study involving the complex geometry of an automotive engine cooling jacket, Laramee et al.[190] generated seed points at the inlet of the cooling jacket. To maintain seed density, a scheme similar to Bauer et al. [142] is adopted and particles travel along integral curves until they hit a boundary or leave through the outlet. Interactively exploring the cooling jacket proved tedious given the rapid

visual clutter created by complicated twisted paths and the difficulty in identifying recirculation zones. To visualize flow past a marine turbine, Peng et al. [191] used derived swirl flow information and multiple-coordinated views to assist domain experts manually place seeds in regions of reverse flow.

To understand and predict the development of cerebral aneurysms, Behrendt et al. [193] proposed an interactive flow visualization technique to isolate pathlines near vessel surfacecs. After domain experts select patches of the surface that match certain features of interest, a pre-computed set of pathlines is filtered in order to retain pathlines within a threshold distance of the surface patch. The resultant pathline bundle then provides the user with a visualization of the local blood flow pattern. Further, the user can specify multiple patches and use color coding to differentiate between pathline bundles.

For a biology-inspired CFD simulation, Koehler et al. [192] presented a novel seed placement method for the visual flow analysis of insect flight. The domain contained multiple dynamically deforming flapping dragonfly wings. Traditional methods of using static seed points suffer in situations where there are immersed boundaries in the flow field. The technique is based on the premise that interesting flow phenomena generally occur near and move with the wings. Seeds are bound in the direction of the vertex normal of user-selected points near the surface of the immersed objects (for this application the wings). The user is given control of the seed density and how far in the normal direction seeds are placed. Seed curves are obtained by connecting points at neighboring time steps that are the same distance in the normal direction of the same point on the wing mesh. Seed curves are then color-mapped to a scalar of interest such as velocity, vorticity or λ_2. The user

can then choose seed curves that are informative and be used further to generate various integration-based flow lines.

9.6 Research Challenges

The majority of automated flow visualization algorithms using integral curves operate under the post hoc visualization paradigm and consider a steady state field. As such, there are many potential solutions (SPSS techniques) available. As of writing this, the most obvious opportunities to improve the existing work are in reducing computation. That said, application of SPSS techniques to unsteady vector fields, in order to visualize integral curves such as pathlines or streaklines in volume flow, still has unsolved problems and continuing challenges. This task is challenging because, unlike streamlines, integral curves in evolving vector fields can intersect in space at different points in time and the amount of vector data to be processed is greater. Further, particles in a unsteady vector field can cluster in certain regions while leaving other regions void. This necessitates tracking of the particle distribution over time to ensure continued coverage of the spatial domain.

In the past decade, multiple works have researched seed placement and stream surfaces selection techniques for the automatic generation of stream surface-based flow visualizations [195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 94]. For stream surfaces, the placement, separation, and alignment of *seed curves* is of interest to researchers. This is an active area of research with new techniques being developed to use stream surfaces effectively.

Uncertainty visualization is an important emerging area of research. Relevant research in the field of uncertain flow visualization has studied the uncertainty that arises in integral curve visualizations from user-input parameters such as seed positions [206]. In another study, Ferstl et al. [207] convey uncertainty

in vector field ensembles using streamline variability plots. Both studies rely on identifying similarity between streamlines in proximity, i.e., they use similarity-based methods, and are applied to steady and unsteady vector fields. Future research could utilize these methods for other integral curves (e.g., streaklines) or to identify optimal instances in time to introduce seeds in the domain.

Another area for the application of SPSS techniques is in situ visualization. In situ processing is typically performed with a limited computational budget and in a distributed memory environment. Initial studies in this space have used SPSS techniques to represent and store steady state vector field data in the form of streamlines [171] and unsteady state vector field data in the form of pathlines [5, 13, 21, 7, 8, 78]. These methods use SPSS techniques to perform an intelligent sampling and reduction of large vector fields such that they can be accurately reconstructed and explored post hoc. Further, the use of SPSS techniques under in situ computational constraints for the purposes of in situ flow visualization, i.e., generating useful flow visualizations as a large-scale simulation progresses, is relatively unexplored.

9.7 Conclusion

This chapter describes the state-of-the-art techniques for seed placement and streamline selection used for flow visualization. The extensive use of streamlines for flow visualization has resulted in several methods and suggested approaches regarding how to use them to explore a flow field. Our classification of these algorithms resulted in three strategy classes (density-based, feature-based, similarity-based) for automatic techniques and two strategy classes (interactive tools, domain information) for manual techniques.

Our survey evaluated automatic techniques to compare and relate them along three axes, namely, redundancy, regions of interest and computation (Figure 46). The three automatic technique classes each offer different benefits. First, density-based techniques provide coverage of the field in either object space or image space and are relatively straightforward. Second, feature-based techniques focus on highlighting salient features of the flow and can often be computed fast. Third, similarity-based techniques minimized redundancy and picked streamlines that together provided the best representative views of the flow. Depending on the use case, different algorithms can be explored or benefits of different algorithms can be combined. Table 17 summarizes the highlights and potential shortcomings of the various SPSS technique categories. Further, Table 16 in the survey shows a grouping of SPSS works based on the context, dimension, and state of flow to which a technique is applied. Although multiple different streamline-based flow visualization tasks have been tackled by SPSS techniques, future research applying these techniques to unsteady state flow, stream/path surface visualization, uncertainty flow visualization, and within an in situ flow visualization context is rich in challenging open problems.

Technique Target/Context	Dims	State	View-Dependent	Distribution	References
Planar Surface Flow	2D	Steady	No	Uniform	[97, 99, 39, 114, 115, 113][15, 132, 133, 93]
	2D	Steady	No	Non-Uniform	[90, 174]
	2D	Steady	Yes	Uniform	[98]
	2D	Steady	Yes	Non-Uniform	[139]
	2D	Unsteady	No	Uniform	[100][131]
Curved Surface Flow	3D	Steady	No	Uniform	[96, 112]
	3D	Steady	Yes	Uniform	[117, 119][130]
Volume Flow	3D	Steady	No	Uniform	[101, 116][125, 128, 129][92]
	3D	Steady	Yes	Uniform	[118, 86, 87, 91, 122][136][89]
	3D	Steady	Yes	Non-Uniform	[120][135][172]
	3D	Steady	No	Non-Uniform	[151, 123, 154, 16, 138, 17, 88, 155][170, 179, 182, 180, 183, 171, 94][189, 188, 190, 191]
	3D	Unsteady	No	Non-Uniform	[137, 141][176][186, 187, 192, 193]
	3D	Unsteady	Yes	Uniform	[121]
Streamsurface Construction	3D	Steady	No	Uniform	[95]
Particle-Based Vis	3D	Unsteady	No	Uniform	[142]
	3D	Steady	No	Non-Uniform	[149]
Texture-Based Vis	3D	Steady	No	Uniform	[104]
	3D	Unsteady	No	Uniform	[105, 106]
	3D	Unsteady	No	Non-Uniform	[144]

Table 16. Grouping of algorithms based on the application context, dimensions, state of flow, dependency on viewpoint, and distribution. References in the right-most column are color-coded based on the type of technique, i.e., density-based (purple), feature-based (blue), similarity-based (red), and manual (green).

Categories of SPSS Techniques	**Highlight**	**Potential Shortcoming**
View-Independent or Object Space	Complete domain coverage.	Often contain redundant streamlines.
View-Dependent or Image Space	Efficient occlusion management, prioritization of interesting streamlines for a given view.	Can contain redundant streamlines.
Flow Topology	Great at emphasizing features.	Can contain redundant streamlines and hard to extend some to 3D.
Derived Field	Good at capturing ROI, fast computation.	Can contain redundant streamlines.
User-Defined Scalar Fields	Can be good at capturing ROI, fast computation.	Not guaranteed to capture ROI.
Spatial Distance	Great at controlling redundancy and capturing ROI.	Large number of distance computations and limited to similarity in proximity.
Shape	Can measure similarity in an orientation-, position-, and scale-invariant manner.	Large preprocessing time for large number of streamlines.
Machine Learning	Great at accurately capturing vector field information in the form of streamlines.	Slow to compute.

Table 17. *A summary of the highlights and potential shortcomings of the various automatic SPSS technique categories. Each category name is colored based on its classification, i.e., purple for density-based, blue for feature-based, and red for similarity-based techniques.*

REFERENCES CITED

[1] J. R. Cash and A. H. Karp, "A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides," *ACM Transactions on Mathematical Software (TOMS)*, vol. 16, no. 3, pp. 201–222, 1990.

[2] K.-L. Ma, "In situ visualization at extreme scale: Challenges and opportunities," *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 14–19, 2009.

[3] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, *et al.*, "In situ methods, infrastructures, and applications on high performance computing platforms," in *Computer Graphics Forum*, vol. 35, pp. 577–597, Wiley Online Library, 2016.

[4] H. Childs, "Data Exploration at the Exascale," *Supercomputing Frontiers and Innovations*, vol. 2, pp. 5–13, Dec. 2015.

[5] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs, "Improved post hoc flow analysis via lagrangian representations," in *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on*, pp. 67–75, IEEE, 2014.

[6] S. Sane and H. Childs, "In situ lagrangian analysis for exploration of time-dependent vector fields," in *ISVFCS20 (accepted for publication)*.

[7] S. Sane, R. Bujack, and H. Childs, "Revisiting the Evaluation of In Situ Lagrangian Analysis," in *Eurographics Symposium on Parallel Graphics and Visualization* (H. Childs and F. Cucchietti, eds.), The Eurographics Association, 2018.

[8] S. Sane, H. Childs, and R. Bujack, "An Interpolation Scheme for VDVP Lagrangian Basis Flows," in *Eurographics Symposium on Parallel Graphics and Visualization* (H. Childs and S. Frey, eds.), The Eurographics Association, 2019.

[9] S. Sane, A. Yenpure, R. Bujack, M. Larsen, K. Moreland, C. Garth, and H. Childs, "Scalable in situ lagrangian flow map extraction: Demonstrating the viability of a communication-free model," *arXiv preprint arXiv:2004.02003*, 2020.

[10] S. Sane, R. Bujack, C. Garth, and H. Childs, "Survey of Seed Placement and Streamline Techniques," *Computer Graphics Forum (accepted for publication)*.

[11] H. Childs, J. Bennett, C. Garth, and B. Hentschel, "In Situ Visualization for Computational Science," *IEEE Computer Graphics and Applications (CG&A)*, vol. 39, pp. 76–85, Nov./Dec. 2019.

[12] J. C. Bennett, H. Childs, C. Garth, and B. Hentschel, "In Situ Visualization for Computational Science (Dagstuhl Seminar 18271)," *Dagstuhl Reports*, vol. 8, pp. 1–43, July 2018.

[13] R. Bujack and K. I. Joy, "Lagrangian representations of flow fields with parameter curves," in *Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on*, pp. 41–48, IEEE, 2015.

[14] M. Hummel, R. Bujack, K. I. Joy, and C. Garth, "Error estimates for lagrangian flow field representations," in *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pp. 7–11, Eurographics Association, 2016.

[15] V. Verma, D. Kao, and A. Pang, "A flow-guided streamline seeding strategy," in *Proceedings of the conference on Visualization'00*, pp. 163–170, IEEE Computer Society Press, 2000.

[16] L. Xu, T.-Y. Lee, and H.-W. Shen, "An information-theoretic framework for flow visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1216–1224, 2010.

[17] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen, "Hierarchical streamline bundles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1353–1367, 2012.

[18] N. A. Petersson and B. Sjögreen, "Wave propagation in anisotropic elastic materials and curvilinear coordinates using a summation-by-parts finite difference method," *Journal of Computational Physics*, vol. 299, pp. 820–841, 2015.

[19] A. Agranovsky, D. Camp, K. I. Joy, and H. Childs, "Subsampling-based compression and flow visualization," in *Visualization and Data Analysis 2015*, vol. 9397, p. 93970J, International Society for Optics and Photonics, 2015.

[20] A. Agranovsky, C. Garth, and K. I. Joy, "Extracting flow structures using sparse particles.," in *VMV*, pp. 153–160, 2011.

[21] J. Chandler, H. Obermaier, and K. I. Joy, "Interpolation-based pathline tracing in particle-based flow visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 1, pp. 68–80, 2015.

[22] G. Haller, "Finding finite-time invariant manifolds in two-dimensional velocity fields," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 10, no. 1, pp. 99–108, 2000.

[23] G. Haller and G. Yuan, "Lagrangian coherent structures and mixing in two-dimensional turbulence," *Physica D: Nonlinear Phenomena*, vol. 147, no. 3-4, pp. 352–370, 2000.

[24] G. Haller, "Distinguished material surfaces and coherent structures in three-dimensional fluid flows," *Physica D: Nonlinear Phenomena*, vol. 149, no. 4, pp. 248–277, 2001.

[25] C. Garth, F. Gerhardt, X. Tricoche, and H. Hans, "Efficient computation and visualization of coherent structures in fluid flow applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1464–1471, 2007.

[26] C. Garth, G.-S. Li, X. Tricoche, C. D. Hansen, and H. Hagen, "Visualization of coherent structures in transient 2d flows," in *Topology-Based Methods in Visualization II*, pp. 1–13, Springer, 2009.

[27] F. Sadlo and R. Peikert, "Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1456–1463, 2007.

[28] F. Sadlo, A. Rigazzi, and R. Peikert, "Time-dependent visualization of lagrangian coherent structures by grid advection," in *Topological Methods in Data Analysis and Visualization*, pp. 151–165, Springer, 2011.

[29] H. Guo, W. He, T. Peterka, H.-W. Shen, S. M. Collis, and J. J. Helmus, "Finite-time lyapunov exponents and lagrangian coherent structures in uncertain unsteady flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 6, pp. 1672–1682, 2016.

[30] T. M. Özgökmen, A. C. Poje, P. F. Fischer, H. Childs, H. Krishnan, C. Garth, A. C. Haza, and E. Ryan, "On multi-scale dispersion under the influence of surface mixed layer instabilities and deep flows," *Ocean Modelling*, vol. 56, pp. 16–30, 2012.

[31] B. Schindler, R. Peikert, R. Fuchs, and H. Theisel, "Ridge concepts for the visualization of lagrangian coherent structures," in *Topological Methods in Data Analysis and Visualization II*, pp. 221–235, Springer, 2012.

[32] M. Hlawatsch, F. Sadlo, and D. Weiskopf, "Hierarchical line integration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 8, pp. 1148–1163, 2011.

[33] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics-theory and application to non-spherical stars," *Monthly notices of the royal astronomical society*, vol. 181, pp. 375–389, 1977.

[34] J. Chandler, R. Bujack, and K. I. Joy, "Analysis of error in interpolation-based pathline tracing," in *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pp. 1–5, Eurographics Association, 2016.

[35] K. E. Atkinson, *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[36] M. Schatzmann, *Numerical Analysis: A Mathematical Introduction*. New York, USA: Oxford University Press, 2002.

[37] N. Brummell, F. Cattaneo, and S. Tobias, "Linear and nonlinear dynamo properties of time-dependent abc flows," *Fluid Dynamics Research*, vol. 28, no. 4, pp. 237–265, 2001.

[38] L. Orf, R. Wilhelmson, and L. Wicker, "Visualization of a simulated Long-Track EF5 tornado embedded within a supercell thunderstorm," *Parallel Comput.*, vol. 0, no. 0, 2015. in press.

[39] A. Mebarki, P. Alliez, and O. Devillers, "Farthest point seeding for efficient placement of streamlines," in *Visualization, 2005. VIS 05. IEEE*, pp. 479–486, IEEE, 2005.

[40] S. C. Shadden, F. Lekien, and J. E. Marsden, "Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows," *Physica D: Nonlinear Phenomena*, vol. 212, no. 3, pp. 271–304, 2005.

[41] S. Waldron, "The error in linear interpolation at the vertices of a simplex," *SIAM Journal on Numerical Analysis*, vol. 35, no. 3, pp. 1191–1200, 1998.

[42] D. Sujudi and R. Haimes, "Integration of particle paths and streamlines in a spatially-decomposed computation," in *Parallel Computational Fluid Dynamics 1995*, pp. 315–322, Elsevier, 1996.

[43] T. Peterka, R. Ross, B. Nouanesengsy, T. Y. Lee, H. W. Shen, W. Kendall, and J. Huang, "A study of parallel particle tracing for steady-state and time-varying flow fields," in *2011 IEEE International Parallel Distributed Processing Symposium*, pp. 580–591, May 2011.

[44] L. Chen and I. Fujishiro, "Optimizing parallel performance of streamline visualization for large distributed flow datasets," in *2008 IEEE Pacific Visualization Symposium*, pp. 87–94, March 2008.

[45] H. Yu, C. Wang, and K.-L. Ma, "Parallel hierarchical visualization of large time-varying 3d vector fields," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, p. 24, ACM, 2007.

[46] B. Nouanesengsy, T.-Y. Lee, and H.-W. Shen, "Load-balanced parallel streamline generation on large scale vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1785–1794, 2011.

[47] J. Dinan, D. B. Larkins, P. Sadayappan, S. Krishnamoorthy, and J. Nieplocha, "Scalable work stealing," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pp. 1–11, IEEE, 2009.

[48] D. Camp, H. Childs, A. Chourasia, C. Garth, and K. I. Joy, "Evaluating the benefits of an extended memory hierarchy for parallel streamline algorithms," in *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pp. 57–64, IEEE, 2011.

[49] C. Muller, D. Camp, B. Hentschel, and C. Garth, "Distributed parallel particle advection using work requesting," in *Large-Scale Data Analysis and Visualization (LDAV), 2013 IEEE Symposium on*, pp. 1–6, IEEE, 2013.

[50] H. Guo, J. Zhang, R. Liu, L. Liu, X. Yuan, J. Huang, X. Meng, and J. Pan, "Advection-based sparse data management for visualizing unsteady flow," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2555–2564, 2014.

[51] D. Pugmire, H. Childs, C. Garth, S. Ahern, and G. H. Weber, "Scalable computation of streamlines on very large datasets," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, p. 16, ACM, 2009.

[52] W. Kendall, J. Wang, M. Allen, T. Peterka, J. Huang, and D. Erickson, "Simplified parallel domain traversal," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 10, ACM, 2011.

[53] K. Lu, H.-W. Shen, and T. Peterka, "Scalable computation of stream surfaces on large scale vector fields," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1008–1019, IEEE Press, 2014.

[54] J. Zhang and X. Yuan, "A survey of parallel particle tracing algorithms in flow visualization," *Journal of Visualization*, vol. 21, no. 3, pp. 351–368, 2018.

[55] R. Bleile, L. Sugiyama, C. Garth, and H. Childs, "Accelerating advection via approximate block exterior flow maps," *Electronic Imaging*, vol. 2017, no. 1, pp. 140–148, 2017.

[56] Y. Liao, H. Matsui, O. Kreylos, and L. H. Kellogg, "Scalable parallel flow visualization using 3d line integral convolution for large scale unstructured simulation data," in *EGPGV*, 2019.

[57] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[58] K. Moreland, C. Sewell, W. Usher, L.-t. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs, *et al.*, "Vtk-m: Accelerating the visualization toolkit for massively threaded architectures," *IEEE Computer Graphics and Applications*, vol. 36, no. 3, pp. 48–58, 2016.

[59] M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci, and C. Harrison, "The alpine in situ infrastructure: Ascending from the ashes of strawman," in *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization*, pp. 42–46, ACM, 2017.

[60] D. Pugmire, A. Yenpure, M. Kim, J. Kress, R. Maynard, H. Childs, and B. Hentschel, "Performance-Portable Particle Advection with VTK-m," in *Eurographics Symposium on Parallel Graphics and Visualization* (H. Childs and F. Cucchietti, eds.), The Eurographics Association, 2018.

[61] A. Fabri and M. Teillaud, "Cgal-the computational geometry algorithms library," in *10e colloque national en calcul des structures*, p. 6, 2011.

[62] W. J. Schroeder, B. Lorensen, and K. Martin, *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[63] D. F. Griffiths and D. J. Higham, *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. London, UK: Springer, 2010.

[64] P. Hartman, "Ordinary differential equations," 1973.

[65] A. Mallinson, D. A. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. A. Jarvis, "Cloverleaf: Preparing hydrodynamics codes for exascale," *The Cray User Group*, vol. 2013, 2013.

[66] S. Popinet, "Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries," *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.

[67] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel, "Nyx: A massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, 2013.

[68] S. K. Lodha, J. C. Renteria, and K. M. Roskin, "Topology preserving compression of 2d vector fields," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 343–350, IEEE, 2000.

[69] S. K. Lodha, N. M. Faaland, and J. C. Renteria, "Topology preserving top-down compression of 2d vector fields using bintree and triangular quadtrees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 4, pp. 433–442, 2003.

[70] H. Theisel, C. Rossl, and H.-P. Seidel, "Combining topological simplification and topology preserving compression for 2d vector fields," in *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, pp. 419–423, IEEE, 2003.

[71] H. Theisel, C. Rössl, and H.-P. Seidel, "Compression of 2d vector fields under guaranteed topology preservation," in *Computer Graphics Forum*, vol. 22, pp. 333–342, Wiley Online Library, 2003.

[72] X. Tong, T.-Y. Lee, and H.-W. Shen, "Salient time steps selection from large scale time-varying data sets with dynamic time warping," in *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 49–56, IEEE, 2012.

[73] U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie, and E. Bethel, "The SENSEI Generic In Situ Interface," in *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV)*, pp. 40–44, IEEE Press, 2016.

[74] T. Fogal, F. Proch, A. Schiewe, O. Hasemann, A. Kempf, and J. Krüger, "Freeprocessing: Transparent in situ visualization via data interception," in *Eurographics Symposium on Parallel Graphics and Visualization*, pp. 49–56, 2014.

[75] M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci, and C. Harrison, "The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman," in *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization (ISAV)*, pp. 42–46, 2017.

[76] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, *et al.*, "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 7, pp. 1453–1473, 2014.

[77] V. Vishwanath, M. Hereld, V. Morozov, and M. E. Papka, "Topology-aware Data Movement and Staging for I/O Acceleration on Blue Gene/P Supercomputing Systems," in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC11)*, pp. 19:1–19:11, 2011.

[78] T. Rapp, C. Peters, and C. Dachsbacher, "Void-and-cluster sampling of large scattered data and trajectories," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 780–789, 2019.

[79] D. Pugmire, T. Peterka, and C. Garth, "Parallel integral curves," *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, pp. 91–113, 2012.

[80] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen, "Over Two Decades of Integration-Based, Geometric Flow Visualization," in *EG 2009 - State of the Art Reports*, pp. 73–92, 2009.

[81] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "The state of the art in flow visualisation: Feature extraction and tracking," in *Computer Graphics Forum*, vol. 22, pp. 775–792, Wiley Online Library, 2003.

[82] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The state of the art in flow visualization: Dense and texture-based techniques," in *Computer Graphics Forum*, vol. 23, pp. 203–221, Wiley Online Library, 2004.

[83] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, "Topology-based flow visualization, the state of the art," in *Topology-based methods in visualization*, pp. 1–19, Springer, 2007.

[84] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matković, and H. Hauser, "The state of the art in topology-based visualization of unsteady flow," in *Computer Graphics Forum*, vol. 30, pp. 1789–1811, Wiley Online Library, 2011.

[85] E. A. Coddington, *An introduction to ordinary differential equations*. Courier Corporation, 2012.

[86] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma, "View-dependent streamlines for 3d vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1578–1586, 2010.

[87] T. Günther, K. Bürger, R. Westermann, and H. Theisel, "A view-dependent and inter-frame coherent visualization of integral lines using screen contribution.," in *VMV*, pp. 215–222, 2011.

[88] J. Ma, J. Walker, C. Wang, S. Kuhl, and C. K. Shene, "Flowtour: An automatic guide for exploring internal flow features," in *Visualization Symposium (PacificVis), 2014 IEEE Pacific*, pp. 25–32, IEEE, 2014.

[89] M. Kanzler, F. Ferstl, and R. Westermann, "Line density control in screen-space via balanced line hierarchies," *Computers & Graphics*, vol. 61, pp. 29–39, 2016.

[90] L. Li, H.-H. Hsieh, and H.-W. Shen, "Illustrative streamline placement and visualization," in *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pp. 79–86, IEEE, 2008.

[91] T. Günther, C. Rössl, and H. Theisel, "Opacity optimization for 3d line fields," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 120, 2013.

[92] Y. Chen, J. Cohen, and J. Krolik, "Similarity-guided streamline placement with error evaluation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1448–1455, 2007.

[93] K. Wu, Z. Liu, S. Zhang, and R. J. Moorhead II, "Topology-aware evenly spaced streamline placement," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 5, pp. 791–801, 2010.

[94] J. Han, J. Tao, and C. Wang, "Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces," *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[95] J. P. M. Hultquist, "Constructing stream surfaces in steady 3d vector fields," in *Proceedings Visualization '92*, pp. 171–178, Oct 1992.

[96] N. Max, R. Crawfis, and C. Grant, "Visualizing 3d velocity fields near contour surfaces," in *Proceedings of the conference on Visualization'94*, pp. 248–255, IEEE Computer Society Press, 1994.

[97] B. Jobard and W. Lefer, "Creating evenly-spaced streamlines of arbitrary density," in *Visualization in Scientific Computing '97*, (Vienna), pp. 43–55, Springer Vienna, 1997.

[98] G. Turk and D. Banks, "Image-guided streamline placement," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 453–460, ACM, 1996.

[99] B. Jobard and W. Lefer, "Multiresolution flow visualization," 2001.

[100] B. Jobard and W. Lefer, "Unsteady flow visualization by animating evenly-spaced streamlines," in *Computer Graphics Forum*, vol. 19, pp. 31–39, Wiley Online Library, 2000.

[101] O. Mattausch, T. Theußl, H. Hauser, and E. Gröller, "Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines," in *Proceedings of the 19th spring conference on Computer graphics*, pp. 213–222, ACM, 2003.

[102] A. Vilanova, G. Berenschot, and C. Van Pul, "Dti visualization with streamsurfaces and evenly-spaced volume seeding," in *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*, pp. 173–182, Eurographics Association, 2004.

[103] D. Merhof, M. Sonntag, F. Enders, P. Hastreiter, R. Fahlbusch, C. Nimsky, and G. Greiner, "Visualization of diffusion tensor data using evenly spaced streamlines," *First publ. in: Vision, Modeling and Visualization / Gnther Greiner ... (eds.). Berlin: Akademische Verl.-Ges. AKA, 2005, pp. 257-264*, 01 2005.

[104] G.-S. Li, U. D. Bordoloi, and H.-W. Shen, "Chameleon: An interactive texture-based rendering framework for visualizing three-dimensional vector fields," in *Visualization, 2003. VIS 2003. IEEE*, pp. 241–248, IEEE, 2003.

[105] H.-W. Shen, U. D. Bordoloi, and G.-S. Li, "Interactive visualization of three-dimensional vector fields with flexible appearance control," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 434–445, 2004.

[106] A. Helgeland and T. Elboth, "High-quality and interactive animations of 3d time-varying vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1535–1546, 2006.

[107] A. Helgeland and O. Andreassen, "Visualization of vector fields using seed lic and volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, no. 6, pp. 673–682, 2004.

[108] L. P. Chew, "Guaranteed-quality mesh generation for curved surfaces," in *Proceedings of the ninth annual symposium on Computational geometry*, pp. 274–280, ACM, 1993.

[109] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[110] H. Edelsbrunner and D. Guoy, "Sink-insertion for mesh improvement," in *Proceedings of the seventeenth annual symposium on Computational geometry*, pp. 115–123, ACM, 2001.

[111] S. Oudot and J.-D. Boissonnat, "Provably good surface sampling and approximation.," in *Symposium on Geometry Processing*, pp. 9–18, 2003.

[112] O. Rosanwo, C. Petz, S. Prohaska, H.-C. Hege, and I. Hotz, "Dual streamline seeding," in *Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific*, pp. 9–16, IEEE, 2009.

[113] W. Zhang, Y. Wang, J. Zhan, B. Liu, and J. Ning, "Parallel streamline placement for 2d flow fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1185–1198, 2013.

[114] W. Zhang, B. Sun, and Y. Wang, "A streamline placement method highlighting flow field topology," in *Computational Intelligence and Security (CIS), 2010 International Conference on*, pp. 238–242, IEEE, 2010.

[115] W. Zhang, M. Zhang, and B. Sun, "Multiresolution streamline placement for 2d flow fields," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pp. 1174–1178, IEEE, 2011.

[116] W. Zhang, J. Ning, M. Zhang, Y. Pei, B. Liu, and B. Sun, "Multiresolution streamline placement based on control grids," *Integrated Computer-Aided Engineering*, vol. 21, no. 1, pp. 47–57, 2014.

[117] X. Mao, Y. Hatanaka, H. Higashida, and A. Imamiya, "Image-guided streamline placement on curvilinear grid surfaces," in *Visualization'98. Proceedings*, pp. 135–142, IEEE, 1998.

[118] L. Li and H.-W. Shen, "Image-based streamline generation and rendering," *IEEE Transactions on Visualization and Computer Graphics*, no. 3, pp. 630–640, 2007.

[119] B. Spencer, R. S. Laramee, G. Chen, and E. Zhang, "Evenly spaced streamlines for surfaces: An image-based approach," in *Computer Graphics Forum*, vol. 28, pp. 1618–1631, Wiley Online Library, 2009.

[120] T. Annen, H. Theisel, C. Rössl, G. Ziegler, and H.-P. Seidel, "Vector field contours," in *Proceedings of Graphics Interface 2008*, pp. 97–105, Canadian Information Processing Society, 2008.

[121] T. Günther, C. Rössl, and H. Theisel, "Hierarchical opacity optimization for sets of 3d line fields," in *Computer Graphics Forum*, vol. 33, pp. 507–516, Wiley Online Library, 2014.

[122] J. Ma, C. Wang, and C.-K. Shene, "Coherent view-dependent streamline selection for importance-driven flow visualization," in *Visualization and Data Analysis 2013*, vol. 8654, p. 865407, International Society for Optics and Photonics, 2013.

[123] L. A. Treinish, "Multi-resolution visualization techniques for nested weather models," in *Proceedings of the conference on Visualization'00*, pp. 513–516, IEEE Computer Society Press, 2000.

[124] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Transactions on Graphics (TOG)*, vol. 5, no. 1, pp. 51–72, 1986.

[125] X. Ye, D. Kao, and A. Pang, "Strategy for seeding 3d streamlines," in *Visualization, 2005. VIS 05. IEEE*, pp. 471–478, IEEE, 2005.

[126] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Saddle connectors-an approach to visualizing the topological skeleton of complex 3d vector fields," in *Visualization, 2003. VIS 2003. IEEE*, pp. 225–232, IEEE, 2003.

[127] K. Mahrous, J. Bennett, G. Scheuermann, B. Hamann, and K. I. Joy, "Topological segmentation in three-dimensional vector fields," *IEEE Transactions on Visualization and Computer Graphics*, no. 2, pp. 198–205, 2004.

[128] Z. Liu, R. Moorhead, and J. Groner, "An advanced evenly-spaced streamline placement algorithm," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 965–972, 2006.

[129] Z. Liu and R. J. Moorhead II, "Robust loop detection for interactively placing evenly spaced streamlines," *Computing in Science & Engineering*, vol. 9, no. 4, pp. 86–91, 2007.

[130] Z. Liu and R. J. Moorhead, "Interactive view-driven evenly spaced streamline placement," in *Visualization and Data Analysis 2008*, vol. 6809, p. 68090A, International Society for Optics and Photonics, 2008.

[131] Z. Ding, X. Zhang, W. Chen, X. Tricoche, D. Peng, and Q. Peng, "Coherent streamline generation for 2-d vector fields," *Tsinghua Science and Technology*, vol. 17, no. 4, pp. 463–470, 2012.

[132] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang, "Vector field editing and periodic orbit extraction using morse decomposition," *IEEE Transactions on Visualization and Computer Graphics*, no. 4, pp. 769–785, 2007.

[133] W. Zhang and J. Deng, "Topology-driven streamline seeding for 2d vector field visualization," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 4901–4905, IEEE, 2009.

[134] W. Zhang and J. Su, "Extraction of limit streamlines in 2d flow field using virtual boundary," in *Computational Intelligence and Security, 2009. CIS'09. International Conference on*, vol. 1, pp. 171–175, IEEE, 2009.

[135] S. Furuya and T. Itoh, "A streamline selection technique for integrated scalar and vector visualization," in *In IEEE Visualization, Poster Session*, 2008.

[136] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis, "View point evaluation and streamline filtering for flow visualization," in *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pp. 83–90, IEEE, 2011.

[137] A. Wiebel and G. Scheuermann, "Eyelet particle tracing-steady visualization of unsteady flow," in *Visualization, 2005. VIS 05. IEEE*, pp. 607–614, IEEE, 2005.

[138] Y. Wang, W. Zhang, and J. Ning, "Streamline-based visualization of 3d explosion fields," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pp. 1224–1228, IEEE, 2011.

[139] C. Luo, I. Safa, and Y. Wang, "Feature-aware streamline generation of planar vector fields via topological methods," *Computers & Graphics*, vol. 36, no. 6, pp. 754–766, 2012.

[140] R. Abraham, J. E. Marsden, and T. Ratiu, *Manifolds, tensor analysis, and applications*, vol. 75. Springer Science & Business Media, 2012.

[141] L. Zhang, G. Chen, R. S. Laramee, D. Thompson, and A. Sescu, "Flow visualization based on a derived rotation field," *Electronic Imaging*, vol. 2016, no. 1, pp. 1–10, 2016.

[142] D. Bauer, R. Peikert, M. Sato, and M. Sick, "A case study in selective visualization of unsteady 3d flow," in *Proceedings of the conference on Visualization'02*, pp. 525–528, IEEE Computer Society, 2002.

[143] D. Stalling and H.-C. Hege, "Fast and resolution independent line integral convolution," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 249–256, ACM, 1995.

[144] S. Guthe, S. Gumhold, and W. Straßer, "Interactive visualization of volumetric vector fields using texture based particles," 2002.

[145] A. Fuhrmann and E. Gröller, "Real-time techniques for 3d flow visualization," in *Proceedings of the conference on Visualization'98*, pp. 305–312, IEEE Computer Society Press, 1998.

[146] J. Kruger, P. Kipfer, P. Konclratieva, and R. Westermann, "A particle system for interactive visualization of 3d flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 6, pp. 744–756, 2005.

[147] K. Burger, P. Kondratieva, J. Kruger, and R. Westermann, "Importance-driven particle techniques for flow visualization," in *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pp. 71–78, Citeseer, 2008.

[148] R. Van Pelt, J. O. Bescos, M. Breeuwer, R. E. Clough, M. E. Groller, B. ter Haar Romenij, and A. Vilanova, "Interactive virtual probing of 4d mri blood-flow," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2153–2162, 2011.

[149] W. Engelke, K. Lawonn, B. Preim, and I. Hotz, "Autonomous particles for interactive flow visualization," in *Computer Graphics Forum*, Wiley Online Library, 2018.

[150] J. Jeong and F. Hussain, "On the identification of a vortex," *Journal of fluid mechanics*, vol. 285, pp. 69–94, 1995.

[151] M. Zockler, D. Stalling, and H.-C. Hege, "Interactive visualization of 3d-vector fields using illuminated stream lines," in *Visualization'96. Proceedings.*, pp. 107–113, IEEE, 1996.

[152] T. Weinkauf and H. Theisel, "Curvature measures of 3d vector fields and their applications," *Journal of WSCG*, vol. 10, pp. 507–514, February 2002.

[153] T. Weinkauf, H.-C. Hege, B. R. Noack, M. Schlegel, and A. Dillmann, "Coherent structures in a transitional flow around a backward-facing step," *Physics of Fluids*, vol. 15, no. 9, pp. S3–S3, 2003.

[154] M. Schlemmer, I. Hotz, B. Hamann, F. Morr, and H. Hagen, "Priority streamlines: A context-based visualization of flow fields.," in *EuroVis*, pp. 227–234, 2007.

[155] H.-W. Shen, R. Vasko, and R. Wenger, "Visualizing flow fields using fractal dimensions," in *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pp. 25–29, Eurographics Association, 2016.

[156] M. Khoury and R. Wenger, "On the fractal dimension of isosurfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1198–1205, 2010.

[157] A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and R. Wenger, "Exploring flow fields using space-filling analysis of streamlines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 10, pp. 1392–1404, 2014.

[158] R. Bujack, J. Kasten, I. Hotz, G. Scheuermann, and E. Hitzer, "Moment invariants for 3d flow fields via normalization," in *2015 IEEE Pacific visualization symposium (PacificVis)*, pp. 9–16, IEEE, 2015.

[159] R. Bujack and H. Hagen, "Moment Invariants for Multi-Dimensional Data," in *Modelling, Analysis, and Visualization of Anisotropy* (E. Ozerslan, T. Schultz, and I. Hotz, eds.), Mathematica and Visualization, Springer Basel AG, 2017.

[160] T. Salzbrunn and G. Scheuermann, "Streamline predicates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1601–1612, 2006.

[161] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer, "Pathline predicates and unsteady flow structures," *The Visual Computer*, vol. 24, no. 12, pp. 1039–1051, 2008.

[162] A. Brun, H. Knutsson, H.-J. Park, M. E. Shenton, and C.-F. Westin, "Clustering fiber traces using normalized cuts," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 368–375, Springer, 2004.

[163] B. Moberts, A. Vilanova, and J. J. Van Wijk, "Evaluation of fiber clustering methods for diffusion tensor imaging," in *Visualization, 2005. VIS 05. IEEE*, pp. 65–72, IEEE, 2005.

[164] L. ODonnell and C.-F. Westin, "White matter tract clustering and correspondence in populations," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 140–147, Springer, 2005.

[165] A. Tsai, C.-F. Westin, A. O. Hero, and A. S. Willsky, "Fiber tract clustering on manifolds with dual rooted-graphs," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–6, IEEE, 2007.

[166] M. Maddah, W. E. L. Grimson, S. K. Warfield, and W. M. Wells, "A unified framework for clustering and quantitative analysis of white matter fiber tracts," *Medical image analysis*, vol. 12, no. 2, pp. 191–202, 2008.

[167] S. Oeltze, D. J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim, "Blood flow clustering and applications invirtual stenting of intracranial aneurysms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 5, pp. 686–701, 2014.

[168] L. Shi, R. S. Laramee, and G. Chen, "Integral curve clustering and simplification for flow visualization: A comparative evaluation," *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[169] L. Shi and G. Chen, "Metric-based curve clustering and feature extraction in flow visualization," 2017.

[170] J. Tao, J. Ma, C. Wang, and C.-K. Shene, "A unified approach to streamline selection and viewpoint selection for 3d flow visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 393–406, 2013.

[171] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang, "Flow field reduction via reconstructing vector data from 3-d streamlines using deep learning," *IEEE Computer Graphics and Applications*, vol. 39, no. 4, pp. 54–67, 2019.

[172] J. Wei, C. Wang, H. Yu, and K.-L. Ma, "A sketch-based interface for classifying and visualizing vector fields," in *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pp. 129–136, IEEE, 2010.

[173] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM (JACM)*, vol. 21, no. 1, pp. 168–173, 1974.

[174] L. Zheng, W. Wang, and S. Li, "Feature-based streamline selection method for 2d flow fields," in *Computer-Aided Design and Computer Graphics (CAD/Graphics), 2015 14th International Conference on*, pp. 129–136, IEEE, 2015.

[175] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, vol. 10, pp. 359–370, Seattle, WA, 1994.

[176] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen, "Similarity measures for enhancing interactive streamline seeding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 8, pp. 1342–1353, 2013.

[177] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[178] C.-K. Chen, S. Yan, H. Yu, N. Max, and K.-L. Ma, "An illustrative visualization framework for 3d vector fields," in *Computer Graphics Forum*, vol. 30, pp. 1941–1951, Wiley Online Library, 2011.

[179] K. Lu, A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and P. C. Wong, "Exploring vector fields with distribution-based streamline analysis.," in *PacificVis*, pp. 257–264, Citeseer, 2013.

[180] Y. Li, C. Wang, and C.-K. Shene, "Streamline similarity analysis using bag-of-features," in *Visualization and Data Analysis 2014*, vol. 9017, p. 90170N, International Society for Optics and Photonics, 2014.

[181] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[182] J. Tao, C. Wang, and C. K. Shene, "Flowstring: Partial streamline matching using shape invariant similarity measure for exploratory flow visualization," in *Visualization Symposium (PacificVis), 2014 IEEE Pacific*, pp. 9–16, IEEE, 2014.

[183] Y. Li, C. Wang, and C.-K. Shene, "Extracting flow features via supervised streamline segmentation," *Computers & Graphics*, vol. 52, pp. 79–92, 2015.

[184] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[185] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[186] S. Bryson, S. Johan, L. Schlecht, B. Green, D. Kenwright, and M. Gerald-Yamasaki, "The virtual windtunnel," in *Computational Fluid Dynamics Review 1998: (In 2 Volumes)*, pp. 1113–1130, World Scientific, 1998.

[187] M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl, "Interactive visualization of fluid dynamics simulations in locally refined cartesian grids (case study)," in *Proceedings of the conference on Visualization'99: celebrating ten years*, pp. 413–416, IEEE Computer Society Press, 1999.

[188] R. S. Laramee, D. Weiskopf, J. Schneider, and H. Hauser, "Investigating swirl and tumble flow with a comparison of visualization techniques," in *Visualization, 2004. IEEE*, pp. 51–58, IEEE, 2004.

[189] R. S. Laramee, "Interactive 3d flow visualization using a streamrunner," in *Conference on Human Factors in Computing Systems: CHI'02 extended abstracts on Human factors in computing systems*, vol. 20, pp. 804–805, 2002.

[190] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen, "Visual analysis and exploration of fluid flow in a cooling jacket," in *Visualization, 2005. VIS 05. IEEE*, pp. 623–630, IEEE, 2005.

[191] Z. Peng, Z. Geng, M. Nicholas, R. S. Laramee, N. Croft, R. Malki, I. Masters, and C. Hansen, "Visualization of flow past a marine turbine: the information-assisted search for sustainable energy," *Computing and Visualization in Science*, vol. 16, no. 3, pp. 89–103, 2013.

[192] C. Koehler, T. Wischgoll, H. Dong, and Z. Gaston, "Vortex visualization in ultra low reynolds number insect flight," *IEEE Transactions on Visualization and Computer Graphics*, no. 12, pp. 2071–2079, 2011.

[193] B. Behrendt, P. Berg, O. Beuing, B. Preim, and S. Saalfeld, "Explorative blood flow visualization using dynamic line filtering based on surface features," in *Computer Graphics Forum*, vol. 37, pp. 183–194, Wiley Online Library, 2018.

[194] D. Sujudi and R. Haimes, "Identification of swirling flow in 3-d vector fields," in *12th Computational Fluid Dynamics Conference*, p. 1715, 1995.

[195] M. Edmunds, T. McLoughlin, R. S. Laramee, G. Chen, E. Zhang, and N. Max, "Automatic stream surface seeding.," in *Eurographics (Short Papers)*, pp. 53–56, 2011.

[196] M. Edmunds, R. S. Laramee, R. Malki, I. Masters, T. Croft, G. Chen, and E. Zhang, "Automatic stream surface seeding: A feature centered approach," in *Computer Graphics Forum*, vol. 31, pp. 1095–1104, Wiley Online Library, 2012.

[197] M. Edmunds, R. S. Laramee, G. Chen, E. Zhang, and N. Max, "Advanced, automatic stream surface seeding and filtering.," in *TPCG*, pp. 53–60, 2012.

[198] J. M. Esturo, M. Schulze, C. Rössl, and H. Theisel, "Global selection of stream surfaces," in *Computer Graphics Forum*, vol. 32, pp. 113–122, Wiley Online Library, 2013.

[199] M. Edmunds, R. S. Laramee, R. Malki, I. Masters, Y. Wang, G. Chen, E. Zhang, and N. Max, "Interactive stream surface placement a hybrid clustering approach supported by tree maps," in *Information Visualization Theory and Applications (IVAPP), 2014 International Conference on*, pp. 347–355, IEEE, 2014.

[200] M. Schulze, J. M. Esturo, T. Günther, C. Rössl, H.-P. Seidel, T. Weinkauf, and H. Theisel, "Sets of globally optimal stream surfaces for flow visualization," in *Computer Graphics Forum*, vol. 33, pp. 1–10, Wiley Online Library, 2014.

[201] M. Bartoň, J. Kosinka, and V. M. Calo, "Stretch-minimising stream surfaces," *Graphical Models*, vol. 79, pp. 12–22, 2015.

[202] A. Brambilla and H. Hauser, "Expressive seeding of multiple stream surfaces for interactive flow exploration," *Computers & Graphics*, vol. 47, pp. 123–134, 2015.

[203] J. Tao and C. Wang, "Peeling the flow: A sketch-based interface to generate stream surfaces," in *SIGGRAPH ASIA 2016 Symposium on Visualization*, p. 14, ACM, 2016.

[204] J. Tao and C. Wang, "Semi-automatic generation of stream surfaces via sketching," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[205] R. Vasko, *Techniques for Assistance in Streamline and Stream Surface Visualizations*. PhD thesis, The Ohio State University, 2017.

[206] T. McLoughlin, M. Edmunds, C. Tong, R. S. Laramee, I. Masters, G. Chen, N. Max, H. Yeh, and E. Zhang, "Visualization of input parameters for stream and pathline seeding," *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, vol. 6, no. 4, pp. 124–135, 2015.

[207] F. Ferstl, K. Bürger, and R. Westermann, "Streamline variability plots for characterizing the uncertainty in vector field ensembles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 767–776, 2015.