

# In Situ Vector Field Reduction via Lagrangian Representations on Supercomputers

Sudhanshu Sane and Hank Childs

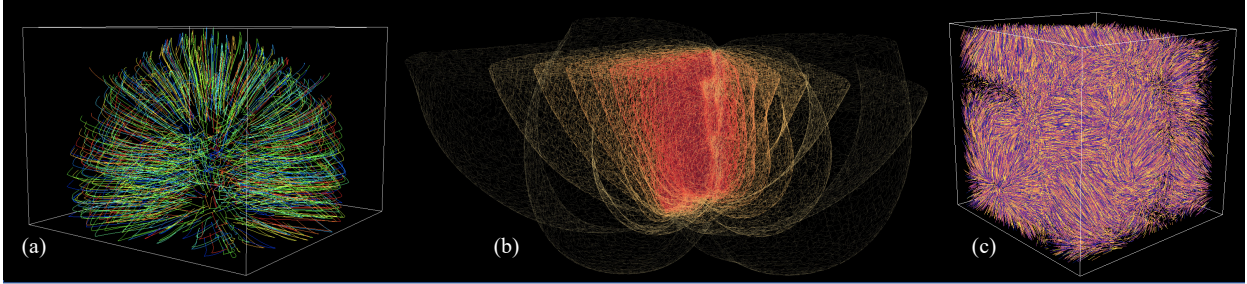


Fig. 1: Visualizations of the simulation codes we demonstrate vector field data reduction for using a Lagrangian representation. L-R: (a) Cloverleaf3D pathlines depicting initial behavior in domain; (b) SW4 displacement magnitude derived from basis trajectories computed over 1000 cycles capturing seismic wave propagation using contour lines; (c) Nyx pathlines for 25 cycles of simulation.

**Abstract**— We conduct an empirical study evaluating *in situ* reduction of time-varying vector field data from a Lagrangian perspective, to enable *post hoc* exploration via flow visualization. This paradigm has the potential to provide significant accuracy and storage advantages over traditional techniques when storage is limited. However, no prior study has demonstrated *in situ* viability. Our study considers state-of-the-art approaches and a variety of workloads, varying over number of particles, interval between outputting data, grid size, and concurrency. It also considers evaluation criteria that span from *in situ* encumbrance to data storage costs to *post hoc* accuracy and performance. The study incorporates integrations with three computational simulations, with 47 experiments on a current top supercomputer. In all, our contribution shows that Lagrangian-based reduction should be the preferred solution for an important class of problems: exploratory time-varying large vector field visualization in I/O constrained settings.

**Index Terms**—In situ processing, scientific visualization, data reduction

## 1 INTRODUCTION

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 the 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 [7]. Algorithms like Runge-Kutta require evaluating the velocity field at multiple locations; the popular fourth order algorithm (RK4) requires evaluation at four locations. 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.

With this work, we consider a class of flow visualization problems with three properties:

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 is saving data to disk at a low rate, i.e., sparse temporal subsampling.

We refer to this as an **EUS** problem: Exploratory analysis + Unsteady-state flow + Sparse 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  $\sim 100X$ , but I/O capabilities have gone up by  $\sim 10X$ . As a result, simulations must reduce proportion of the data they store, often creating **Sparse** 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 [5, 26] 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,

• Sudhanshu Sane is with the University of Oregon, USA. E-mail: ssane@uoregon.edu.

• Hank Childs is with the University of Oregon, USA. E-mail: hank@uoregon.edu.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxxx

this *in situ* style is not congruent with the Exploratory component of **EUS** problems. Fortunately, an alternate workflow avoids the need for *a priori* knowledge, by using a combination of *in situ* and *post hoc* processing [10]. 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**: Lagrangian-based In Situ Reduction with Post Hoc Exploration.

Agranovsky et al. [1] performed the seminal study on applying **L-ISR-PHE** to the **EUS** problem, but their study left significant questions. In particular, while their work demonstrated promising accuracy-storage tradeoffs, their work failed to seriously consider whether the approach could be practically deployed. Further, follow-on studies have also not considered practical deployment, instead focusing on better understanding accuracy-storage tradeoffs or improving *post hoc* interpolation. As a result, a significant open question is whether the **L-ISR-PHE** approach is practical. The open nature of this question has hindered both follow-on research and widespread adoption.

The contribution of this paper is 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.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Frames of Reference

In fluid dynamics, there are two frames of reference to observe fluid motion: Eulerian and Lagrangian. With the Eulerian frame of reference, the observer is in a fixed position. With the Lagrangian frame of reference, the observer is attached to a fluid parcel and is moving through space and time.

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  $t \in \mathbb{R}$

$$v : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d, x, t \mapsto v(x, t) \quad (1)$$

In a practical setting, a flow field at a specific time/cycle is defined as a vector data on a fixed, discrete mesh. Time-varying flow is represented as a collection of such data over a variety times/cycles.

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 is comprised of the starting positions of massless particles  $x_0$  at time  $t_0$  and their respective trajectories that are interpolated using the time-dependent vector field.

Mathematically, a flow map is defined as the mapping

$$F_{t_0}^t(x_0) : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d \rightarrow \mathbb{R}^d, t \times t_0 \times x_0 \mapsto F_{t_0}^t(x_0) = x(t) \quad (2)$$

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

$$\frac{d}{dt}x(t) = v(x(t), t) \quad (3)$$

In a practical setting, the flow map is represented 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 (i.e., a Lagrangian representation), encodes the behavior of the time-dependent vector field over an interval of time.

Although the frames of reference are theoretically equivalent [6], their application in practical settings varies. Our interest with this study is the practical setting, specifically for the **EUS** problem.

### 2.2 In Situ Reduction via Lagrangian Representations

As referenced in the introduction, Agranovsky et al. [1] presented the first work on applying **L-ISR-PHE** to **EUS**. Their work compared to a traditional Eulerian approach, i.e., saving cycles at regular intervals and calculating pathlines using temporal interpolation. Their findings demonstrated significant benefits, for example 12X improvements in accuracy using the same storage, as well as the same accuracy for 64X less storage. The key intuition behind these results is that the Lagrangian representation captures the behavior of the flow field over an interval of time, as opposed to the state at a single time slice, and thus is able to store more information per byte with respect to **EUS**.

Subsequent research studies have broadened the understanding and exploration of the **L-ISR-PHE** paradigm. Sane et al. [35] evaluated the **L-ISR-PHE** workflow for a range of spatiotemporal configurations operating on a fixed storage budget and provided absolute error estimates. Most recent research has explored sampling strategies for particle trajectories that form the Lagrangian representation. Sane et al. [36] explored the use of longer trajectories and proposed an interpolation scheme to reduce error by evaluating neighborhoods across interpolations. Rapp et al. [32] applied their void-and-cluster sampling technique to identify a representative set of scattered particles and found that it performs better than random sampling. To address scalability challenges, Sane et al. [37] explored an accuracy-performance tradeoff and demonstrated the use of a communication-free model that only stored trajectories that remain within the rank domain during the interval of computation.

### 2.3 Lagrangian Extract-Based Post Hoc Exploration

The notion of calculating particle trajectories for a scientific simulation “online” for “offline” exploration was first explored by Vries et al. [43] in the context of ocean circulation models. Next, Hlawatsch et al. [19] proposed a pathline interpolation technique that employs a hierarchical scheme and assumes access to the data across multiple time steps. The technique constructs longer, more accurate trajectories by decreasing the number of integration steps when using previously computed particle trajectories. Chandler et al. [9] proposed a modified k-d tree as a search structure and an interpolation technique for Lagrangian data extracted from an SPH simulation. Further, Chandler et al. [8] conducted error analysis studies and identified correlations between Lagrangian-based interpolation error and divergence in the flow field. Bujack et al. [6] evaluated the use of parameter curves to fit interpolated pathline points to improve the aesthetic of trajectories calculated using Lagrangian data. Hummel et al. [20] provided theoretical error bounds for error propagation and accumulation that can occur when calculating pathlines using Lagrangian data. With respect to adoption of **L-ISR-PHE**, Agranovsky et al. [1]’s introduced a scheme, although their interpolation was straightforward since their *in situ* scheme place particles at uniform locations and terminating them at regular intervals. Recently, Pascal et al. [29, 38] used extracted Lagrangian data from an ocean modeling simulation to explore coastal upwelling activity and visualize a derived scalar field representing pathline density.

### 2.4 Other Relevant Research

Within the vector field analysis and visualization community, Lagrangian methods have been increasingly used in the past decade. Lagrangian coherent structures (LCS) are a popular technique to visualize attracting and repelling surfaces and were introduced by Haller et al [16–18]. The interest in the technique led to multiple efforts that were aimed at accelerating the computation and visualization of LCS [13, 14, 33, 34]. LCS have also been used for uncertain transient vector field visualization by Guo et al. [15].

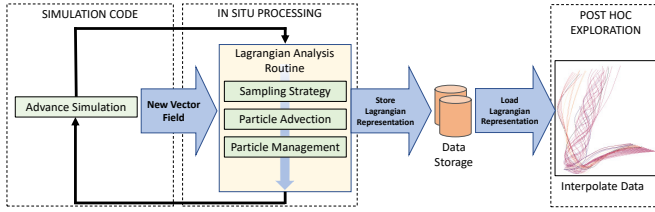


Fig. 2: Schematic diagram of the **L-ISR-PHE** workflow showing *in situ* processing, data storage, and *post hoc* exploration.

Finally, multiple works have proposed vector field reduction strategies while maintaining an Eulerian representation. Lodha et al. [25] controlled the compression of similar vectors into single vectors representing larger area. Further, Lodha et al. [24] proposed a top-down topology preserving compression technique. Theisel et al. [39] computed critical points and viewed the task as a mesh reduction, and later provided a threshold to filter important features [40]. 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. [41] 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 don't inherently address the challenge of increasing temporal sparsity.

### 3 EVALUATING L-ISR-PHE

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

- Subsection 3.1 describes the instantiation we consider.
- Subsection 3.2 describes evaluation criteria.
- Subsection 3.3 describes important factors in defining workloads.

#### 3.1 Instantiation

Figure 2 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:** *In situ* reduction operates by maintaining a set of particles and their trajectories. As the simulation completes each cycle, it then invokes our visualization routines to update particle positions to reflect the advancement in time. The main decisions for our visualization routines are when and where to introduce particles, when to terminate particles, and what information to save about particles. For this empirical study, we introduced particles using the uniform seed placement scheme from Agranovsky et al. [1], including re-introducing particles at fixed intervals. Our particle termination follows the communication-free model from Sane et al. [37], 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:** Lowering the encumbrance on file data storage systems is one of the primary motivators to use the **L-ISR-PHE** workflow. Data storage costs can vary based on 1) how many particles are used to sample the domain, and 2) what information is stored for each particle. Typically, when using more particles, and thus more data storage, a stakeholder would expect greater integrity during exploration. Further, the efficacy of exploration can be closely tied to the nature of the stored data. Although the general **L-ISR-PHE** framework can support complex Lagrangian representations (unstructured particle sets, polynomial expressions, attribute information, etc.), these are yet unexplored. 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 [1, 35, 37]. These saved particle trajectories are called “basis flows.”

**Post Hoc Exploration:** The final phase of the **L-ISR-PHE** workflow involves exploratory flow visualization, which in turn requires constructing new particle trajectories. These new particles trajectories

are constructed by interpolating from the basis flows. As mentioned in the discussion of storage, the efficacy of the technique is dependent on the nature of the data stored. Prior works have looked extensively at the theoretical and empirical error of Lagrangian-based advection schemes [6, 8, 19, 20, 35] and proposed methods to interpolate long trajectories [36] and particles with non-zero mass [9]. 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 by Sane et al. [37], 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. [2].

#### 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 [4, 12, 22, 23, 42], 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 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.

#### 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**

## 4 EMPIRICAL STUDY OVERVIEW

This section provides an overview of our experiments. It is organized as follows: experiments performed (4.1), simulation codes (4.2), software implementation (4.3), hardware (4.4), and metrics (4.5).

### 4.1 Experiment Overview

Our experiments are designed in response to the evaluation “areas” from Section 3.1. 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 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 1 shows our choices for the *in situ* encumbrance campaign, while Table 2 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 1: 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 2: Experimental overview for the *post hoc* efficacy campaign. Overall, we considered 25 experiments for this campaign.

## 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 [27] 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 a show pathlines calculated in the Cloverleaf3D domain that show this initial behavior in the simulation.

Next, we consider the SW4 seismology simulation [30]. 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 b 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 [3], 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.

## 4.3 Software Implementation

### 4.3.1 In Situ Reduction

We use the Ascent [21] *in situ* infrastructure and VTK-m [28] 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 [31]. 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 double), 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.

### 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 triangulation over the start locations of valid basis particles using CGAL [11] (parallel via TBB) 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.



#### 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 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.

#### 4.5 Evaluation Metrics

##### 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  $\text{Sim}_{\text{cycle}}$ . 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.

##### 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 infrequently in our case, we observed for the scale of study we conducted that Summit provides very fast write times. Table 3 documents write times of varying file sizes in binary format on Summit. Given the range of

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 3: 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.

file sizes stored to disk by 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.

##### 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  $\text{Avg}_{L2}$  and  $\text{Max}_{L2}$  to denote the average and maximum L2-norm for an individual particle trajectory, respectively.

An overall average of  $\text{Avg}_{L2}$ , denoted by  $\text{AvgN}_{L2}$ , is measured across  $N$  test particle samples and provides a robust statistic [1, 32, 35, 36]. Unlike  $\text{Avg}_{L2}$ , 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 ( $\text{Avg}_{L2}$ ,  $\text{Max}_{L2}$ ) across all test particles and provides insight into per particle outcomes. Further, for **PHE-2** we include a study of *post hoc* reconstruction costs.

## 5 RESULTS AND DISCUSSION

We evaluated **L-ISR-PHE** using seismic wave propagation, cosmology, and proxy ECP hydrodynamics simulations (Section 4.2) for two aspects: (1) *in situ* encumbrance under varying workloads (Section 5.1); and (2) *post hoc* efficacy for various configurations (Section 5.2).

### 5.1 In Situ Encumbrance

Table 4 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 4 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 4.

#### 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  $\text{Sim}_{\text{cycle}}$  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  $\text{Sim}_{\text{cycle}}$  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  $\text{Sim}_{\text{cycle}}$  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**.

#### 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  $\text{Sim}_{\text{cycle}}$ . 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  $\text{Sim}_{\text{cycle}}$  is very small (0.035s) in this case. In contrast, for the largest grid size (each rank operated on

Nodes	MPI Ranks	Dimensions	Interval	Sim <sub>cycle</sub>	Particles /Node	Memory /Node (MB)	Step	DAV%	Scatter Plots																																	
Cloverleaf3D Proxy Hydrodynamics Application									<div>6 GPUs/Node particle advection <b>Step</b> times Markers: Cloverleaf3D (orange), SW4 (blue)</div> <table><thead><tr><th>Number of Particles/Node (x1000)</th><th>In Situ Step Time (s)</th><th>Application</th></tr></thead><tbody><tr><td>128</td><td>0.015</td><td>SW4</td></tr><tr><td>256</td><td>0.10</td><td>Cloverleaf3D</td></tr><tr><td>256</td><td>0.11</td><td>Cloverleaf3D</td></tr><tr><td>256</td><td>0.12</td><td>Cloverleaf3D</td></tr><tr><td>512</td><td>0.16</td><td>Cloverleaf3D</td></tr><tr><td>512</td><td>0.17</td><td>Cloverleaf3D</td></tr><tr><td>1024</td><td>0.32</td><td>Cloverleaf3D</td></tr><tr><td>1024</td><td>0.35</td><td>Cloverleaf3D</td></tr><tr><td>1024</td><td>0.42</td><td>Cloverleaf3D</td></tr><tr><td>2048</td><td>0.35</td><td>SW4</td></tr></tbody></table>	Number of Particles/Node (x1000)	In Situ Step Time (s)	Application	128	0.015	SW4	256	0.10	Cloverleaf3D	256	0.11	Cloverleaf3D	256	0.12	Cloverleaf3D	512	0.16	Cloverleaf3D	512	0.17	Cloverleaf3D	1024	0.32	Cloverleaf3D	1024	0.35	Cloverleaf3D	1024	0.42	Cloverleaf3D	2048	0.35	SW4
Number of Particles/Node (x1000)	In Situ Step Time (s)	Application																																								
128	0.015	SW4																																								
256	0.10	Cloverleaf3D																																								
256	0.11	Cloverleaf3D																																								
256	0.12	Cloverleaf3D																																								
512	0.16	Cloverleaf3D																																								
512	0.17	Cloverleaf3D																																								
1024	0.32	Cloverleaf3D																																								
1024	0.35	Cloverleaf3D																																								
1024	0.42	Cloverleaf3D																																								
2048	0.35	SW4																																								
16	96	586 × 586 × 586	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 × 251 × 70	200	0.35	555k	13.89	0.0412	11.67																																		
		335 × 335 × 93		2.02	1.3M	33.16	0.2125	10.48																																		
		501 × 501 × 139		7.58	4.4M	111.13	0.3309	4.365																																		
64	384	1001 × 1001 × 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																																			
		2.9		1.2M	31.9	0.2095	7.074																																			
		Nyx Cosmology Simulation																																								
1	1	65 × 65 × 65	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 × 129 × 129		88.3	2.1M	53.6	0.0596	0.067																																		
					262k	6.5	0.0101	0.011																																		
					32k	0.8	0.0044	0.005																																		

2 CPUs/Node particle advection **Step** times  
Markers: Nyx 65^3 (cyan), Nyx 128^3 (red)

Number of Particles/Node (x1000)	In Situ Step Time (s)	Application
16	0.002	Nyx 65^3
32	0.003	Nyx 65^3
32	0.004	Nyx 128^3
128	0.008	Nyx 128^3
256	0.012	Nyx 65^3
256	0.013	Nyx 128^3

Table 4: *In situ* encumbrance evaluation and experiment configurations for our three simulation codes.

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<sub>cycle</sub> 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<sub>cycle</sub> with small variability. However, as we increased the grid size, and consequentially, the workload from 540k to 1.2M particles per node (~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<sub>cycle</sub>. Although extracting a Lagrangian representation might place a higher encumbrance on a simulation with a small Sim<sub>cycle</sub> 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.

### 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<sub>cycle</sub> 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).

Overall, considering the larger Sim<sub>cycle</sub> 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.

## 5.2 Post Hoc Efficacy

Table 5 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 Side%** column in Table 5 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 5, 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

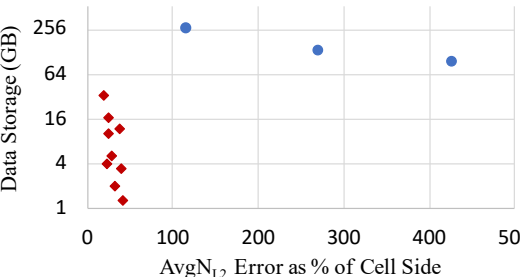
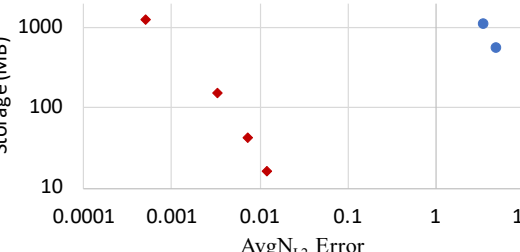
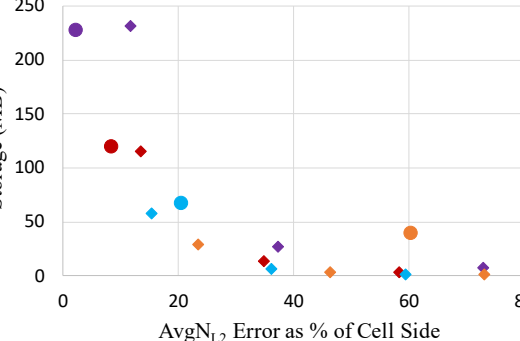
Technique	Interval	Reduction	Data	AvgN <sub>L2</sub>	Cell Side%	Scatter Plot
Cloverleaf3D Proxy Hydrodynamics Application						<p>Cloverleaf 3D Accuracy-Storage</p> <p>Markers: Lagrangian (red), Eulerian (blue)</p> 
Eulerian	20	Full Res	267 GB	0.0197	116.17	
	40		133 GB	0.0459	270.49	
	60		95 GB	0.0725	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						<p>SW4 Accuracy-Data Storage</p> <p>Markers: Lagrangian (red), Eulerian (blue)</p> 
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						<p>Nyx Accuracy-Data Storage</p> <p>Encoding: Lagrangian (square), Eulerian (circle)</p> <p>Interval 25 (purple), Interval 50 (red), Interval 100 (cyan), Interval 200 (orange)</p> 
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 5: *Post hoc* efficacy evaluation and experiment configurations for our three simulation codes.

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.

### 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 Avg $N_{L2}$  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 Avg $N_{L2}$  of 270.4% of the cell side. For all the Lagrangian configurations, the Avg $N_{L2}$  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 3 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 **Max $N_{L2}$**  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.

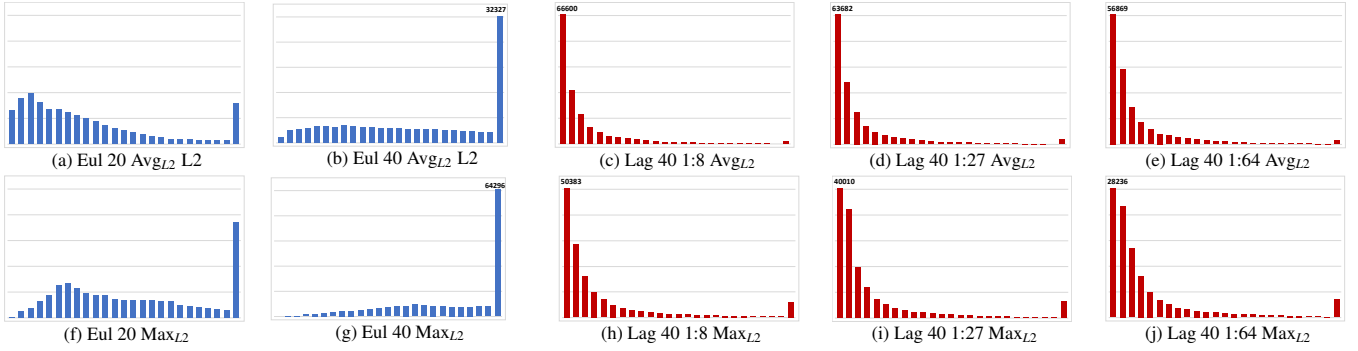


Fig. 3: **Cloverleaf3D** experiment histograms for 100,000 test particle interpolation errors. Each plot has 25 bins, ranging from 0 to  $>0.05$ , with bar height encoding number of particles. Horizontal grid lines mark increments of 5,000.

Samples /Rank	CGAL Delaunay (s)	Interpolation (s)	Communication (s)
7.2M	24.6	0.00246	
2.1M	7.05	0.00141	0.00125
887k	2.89	0.00093	

Table 6: 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.

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 6 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 low compared to the Eulerian approach that requires performing interpolation and communication for every cycle. For example, the Lagrangian configuration using an interval of 40 and 1:27 number of particles, can be used to construct pathlines for 100,000 particles across 600 cycles in under 2 minutes (excluding I/O). For an Eulerian approach to be faster, it would need to compute each cycle in 0.2 seconds (our implementation required 0.47 seconds per cycle excluding I/O).

### 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  $\text{AvgN}_{L2}$  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 5) using 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.

### 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 4 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.

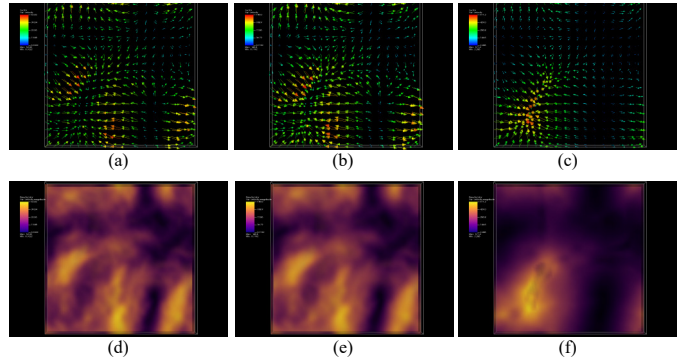


Fig. 4: Nyx vector field visualization: (a) and (d) show the vector field at time 0 and the maximum velocity magnitude is 52.02, (b) and (e) show the vector field at time 200 and the maximum velocity magnitude is 145.0, and finally, (c) and (f) show the vector field at time 400 and the maximum velocity 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 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 4). 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 [6, 19, 20, 36].

As the interval size increases, i.e., the Sparsity 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 6. Of course, a longer interval does not guarantee better accuracy for the Lagrangian technique in all settings. Divergence over a long interval impacts Lagrangian-based interpolation accuracy [8].

With respect to the impact of number of particles on **PHE-1** and



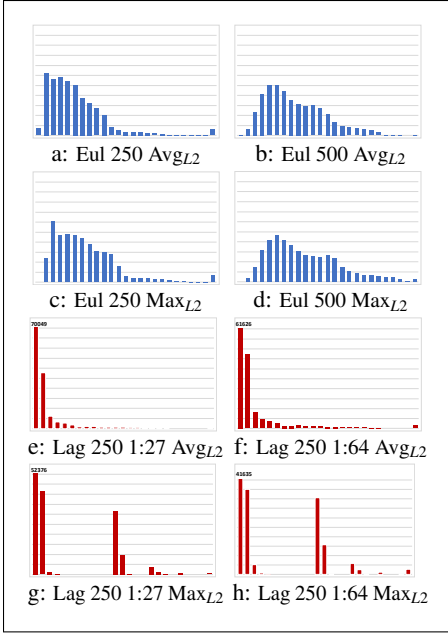


Fig. 5: **SW4** experiment histograms for 90,000 test particle interpolation errors. Each plot has 25 bins, Eulerian bins range from  $<0.6$  to  $>15$ , Lagrangian bins range from 0 to  $>0.2$ , with bar height encoding number of particles. Horizontal grid lines mark increments of 2,000.

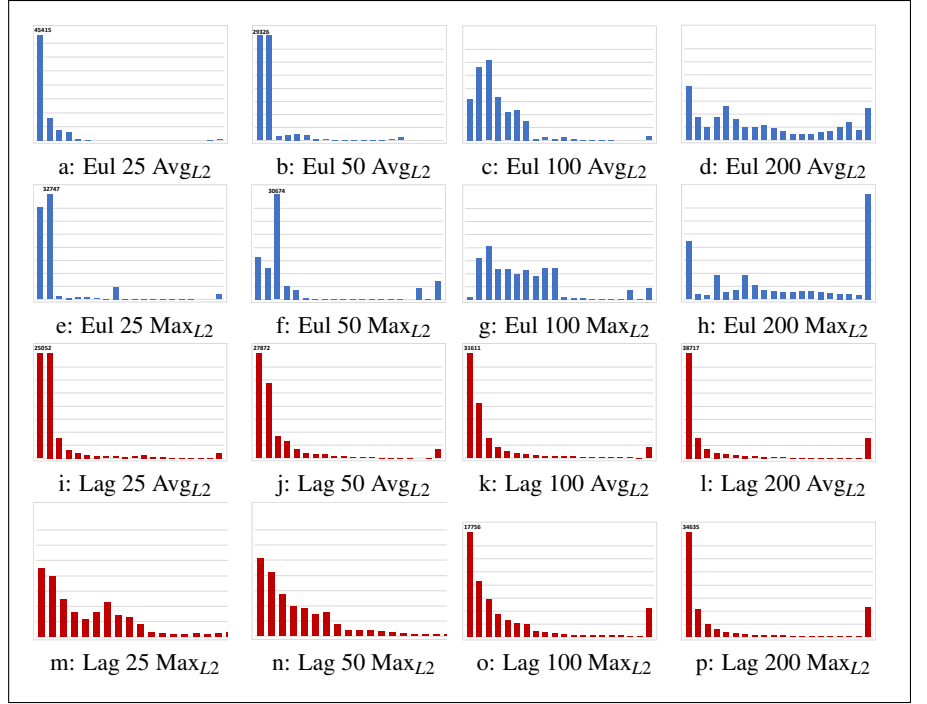


Fig. 6: **Nyx** experiment histograms for 50,000 test particle interpolation errors. Each plot has 20 bins, ranging from 0 to  $>0.44$ , with bar height encoding number of particles. Horizontal grid lines mark increments of 2,000.

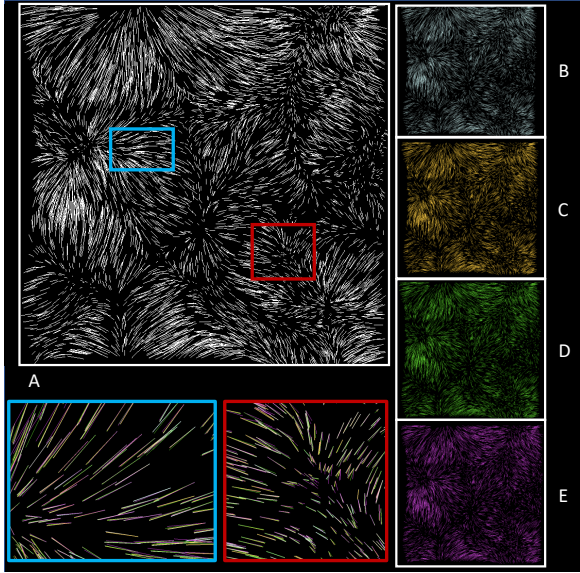


Fig. 7: A qualitative comparison of pathline reconstruction over one interval for the Nyx data (**Interval = 25** case). For each configuration we specify **DS-1** and **PHE-1** as (Bytes, Cell Side%). Image A (white) is the ground truth, B (light-blue) is Eulerian (227MB; 2.29%), C (yellow) is Lagrangian 1:1 (232MB; 11.61%), D (green) is Lagrangian 1:8 (27MB; 37.27%), E (pink) is Lagrangian 1:27 (8MB; 72.72%).

**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  $\text{AvgN}_{L2}$  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.

## 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. In turn, we hope the study enables additional adoption from stakeholders and future research on improved techniques. In terms of future work, we believe that research on *in situ* extraction could lead to even better accuracy-storage tradeoffs, especially with respect to particle placement and termination, and with respect to increased resolution along a trajectory.

## ACKNOWLEDGMENTS

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

## REFERENCES

- [1] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs. Improved Post Hoc Flow Analysis via Lagrangian Representations. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 67–75, Nov. 2014.
- [2] 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.
- [3] 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*, 765(1):39, 2013.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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)*, 16(3):201–222, 1990.
- [8] 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.
- [9] J. Chandler, H. Obermaier, and K. I. Joy. Interpolation-based pathline tracing in particle-based flow visualization. *IEEE transactions on visualization and computer graphics*, 21(1):68–80, 2015.
- [10] H. Childs. Data Exploration at the Exascale. *Supercomputing Frontiers and Innovations*, 2(3):5–13, Dec. 2015.
- [11] A. Fabri and M. Teillaud. Cgal-the computational geometry algorithms library. In *10e colloque national en calcul des structures*, p. 6, 2011.
- [12] 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.
- [13] 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*, 13(6):1464–1471, 2007.
- [14] 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.
- [15] 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*, 22(6):1672–1682, 2016.
- [16] G. Haller. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 10(1):99–108, 2000.
- [17] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena*, 149(4):248–277, 2001.
- [18] G. Haller and G. Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena*, 147(3-4):352–370, 2000.
- [19] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Hierarchical line integration. *IEEE transactions on visualization and computer graphics*, 17(8):1148–1163, 2011.
- [20] 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.
- [21] 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 for Enabling Extreme-Scale Analysis and Visualization*, pp. 42–46. ACM, 2017.
- [22] 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 for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, pp. 42–46, 2017.
- [23] 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*, 26(7):1453–1473, 2014.
- [24] S. K. Lodha, N. M. Faaland, and J. C. Renteria. Topology preserving top-down compression of 2d vector fields using bintree and triangular quadrees. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):433–442, 2003.
- [25] 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.
- [26] K.-L. Ma. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, 2009.
- [27] 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*, 2013, 2013.
- [28] 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*, 36(3):48–58, 2016.
- [29] P. Nardini, M. Bttinger, G. Scheuermann, and M. Schmidt. Visual Study of the Benguela Upwelling System using Pathline Predicates. In K. Rink, A. Middel, D. Zeckzer, and R. Bujack, eds., *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2017. doi: 10.2312/envirvis.20171099
- [30] 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*, 299:820–841, 2015.
- [31] D. Pugmire, A. Yenpure, M. Kim, J. Kress, R. Maynard, H. Childs, and B. Hentschel. Performance-Portable Particle Advection with VTK-m. In H. Childs and F. Cucchietti, eds., *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2018. doi: 10.2312/pgv.20181094
- [32] T. Rapp, C. Peters, and C. Dachsbacher. Void-and-cluster sampling of large scattered data and trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):780–789, 2019.
- [33] F. Sadlo and R. Peikert. Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1456–1463, 2007.
- [34] 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.
- [35] S. Sane, R. Bujack, and H. Childs. Revisiting the Evaluation of In Situ Lagrangian Analysis. In H. Childs and F. Cucchietti, eds., *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2018. doi: 10.2312/pgv.20181096
- [36] S. Sane, H. Childs, and R. Bujack. An Interpolation Scheme for VDVP Lagrangian Basis Flows. In H. Childs and S. Frey, eds., *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2019. doi: 10.2312/pgv.20191115
- [37] 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.
- [38] L. Siegfried, M. Schmidt, V. Mohrholz, H. Pogrzeba, P. Nardini, M. Böttinger, and G. Scheuermann. The tropical-subtropical coupling in the southeast atlantic from the perspective of the northern benguela upwelling system. *PloS one*, 14(1), 2019.
- [39] H. Theisel, C. Rossl, and H.-P. Seidel. Combining topological simplification and topology preserving compression for 2d vector fields. In *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.*, pp. 419–423. IEEE, 2003.
- [40] 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.
- [41] 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.

- [42] 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.
- [43] P. Vries and K. Döös. Calculating lagrangian trajectories using time-dependent velocity fields. *Journal of Atmospheric and Oceanic Technology*, 18(6):1092–1101, 2001.