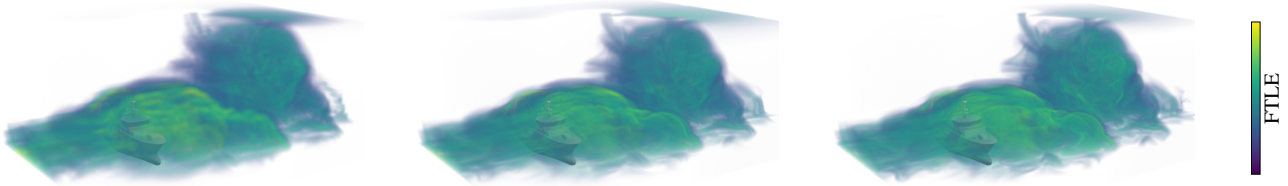
Eurographics Conference on Visualization (EuroVis) 2026
N. Andrienko, M. Chen, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 3

Anchor Flow Maps: Efficient Sampling and Approximation of the Entire Flow Map

D. Stelter, T. Wilde, C. Rössl and H. Theisel

University of Magdeburg, Germany



(a) *FFM, res_F* **(b)** *AFM_c, res_A* **(c)** *Integrated*

Figure 1: *Volume renderings of the finite-time Lyapunov exponent (FTLE) for the Tangaroa dataset with $t_0 = 0.8$, $t_1 = 1.2$. FTLE is computed using samples of the flow map. In (a), these samples are taken from a naive 5D full flow map (FFM) sampling with resolution res_F. In (b), we apply our anchor flow map (AFM) approach with the much larger resolution res_A at same memory consumption. This results in a more detailed FTLE computation that is closer to the actual FTLE field, see direct pathline integration in (c).*

Abstract

Flow maps are a fundamental concept for the visualization and analysis of time-dependent flows. A common approach is to apply sampling and reconstruction – which comes with a number of problems: flow maps are high-dimensional functions with locally large gradients which makes approximations very challenging. In general, one must apply large sampling resolutions and more advanced methods to achieve sufficiently good results. Existing methods for efficient, high-quality sampling and approximation restrict to subsets of flow maps, e.g., with fixed start and end times, or at least fixed differences. In this paper, we propose the first method to handle the entire flow map by decreasing the dimensionality of the space-time-time domain. Essentially, it is a framework which splits the high-dimensional flow map into few lower-dimensional functions. This enables to achieve much larger resolutions than traditional sampling with the same memory consumption. We use our method in established applications and show that it can significantly decrease errors in the reconstruction by using these larger resolutions.

CCS Concepts

• ***Human-centered computing*** → *Scientific visualization;* • ***Computing methodologies*** → *Computer graphics;*

1. Introduction

In many scientific fields, flow visualization is an essential approach for analyzing complex fluid dynamics. Example domains are aerodynamics, meteorology, oceanography, and medical imaging. There exists a variety of methods, all concerned with different properties and flow features. Each of these approaches aim to make the flows interpretable to human experts. One can represent flows in different ways. The most common approach is to use unsteady velocity fields which describe the flow in an Eulerian way. Thus, it provides the current advection for any position and time point. A multitude of methods has been proposed for their visualization. However, many visualization techniques require pathlines which are integral curves of velocity fields over time. Therefore, pathlines represent the trajectories of massless particles. They are computed through numerical integration which is computationally expensive.

Even though velocity fields still are an important research topic in flow visualization, more recent research focussed another form of flow representation: the flow map. It describes the flow in the Lagrangian way and encodes the information of all pathlines present. This works by mapping the start position, start time and end time to the destination position of a massless particle which has been advected by the flow. This means that the flow map has a space-time-time domain and maps to another position in space. Flow maps provide an elegant way to describe the behavior of all massless particles in a flow. However, considering the flow map is still a challenging task because of three problems:

1. The computation of flow maps is expensive, as it involves a massive amount of numerical particle integrations.
2. Flow maps tend to have strong variations that manifest by strong gradients of the flow map field. In fact, flow map gradients can grow exponentially with integration time in areas of Lagrangian flow separation.
3. Flow maps are high-dimensional: the flow map of a 2D flow is a 4D field, and the flow map of a 3D flow is even a 5D field.

In recent years, visualization and other communities have carried out extensive research to address these problems. To cope with problem 1, several approaches exist to pre-compute the flow map at certain samples, and to compute missing flow maps by interpolation or learning-based approaches from the pre-computed ones. Instead of numerically integrating flow maps on-the-fly, a sampling and subsequent reconstruction of the flow map is done. Problem 2 is usually addressed by applying adaptive flow map sampling techniques, allowing higher sampling density in areas of higher flow map gradients.

Existing approaches for problem 3 are surprisingly simple: they do not consider the entire high-dimensional flow map but only a subset. To the best of our knowledge, all existing approaches for flow map sampling fix the start time, the integration duration, or both. This way, the dimensionality of the flow map is reduced to a manageable amount of data, at the expense that only a small part of the flow map is considered at all.

The research presented in this paper is guided by the conviction that a comprehensive visual analysis of a flow requires the consideration of the entire space-time-time domain of the flow map. We present an approach to sample the entire flow map. Our main theoretical contribution is to show that the high-dimensional flow map can be completely represented by three lower-dimensional fields. This allows us to sample and reconstruct these lower-dimensional fields even in cases where a naive sampling of the entire flow map is insufficient due to memory constraints. Additionally, we propose an extension of the theoretical concept to additionally compute further Lagrangian information. More specifically, we include the efficient approximation of the Lagrangian-averaged vorticity deviation (LAVD) to our framework. We discuss the numerical accuracy and computational efficiency of our methods, and apply them to compute results of the entire flow map in real-time. Based on this, we present interactive visualizations of the finite-time Lyapunov exponent (FTLE) and LAVD fields as two standard representatives of Lagrangian approaches for the entire space-time-time domain.

2. Related Work

2.1. Applications of the Flow Map

Over the last decades, researchers proposed a variety of applications of the flow map. Finite-time Lyapunov exponents (FTLE) are among the most common. FTLE extracts Lagrangian coherent structures (LCS) which describe the topological skeleton of time-dependent flows [HY00, Hal01, SLM05]. A fundamental approach was presented by Sadlo and Peikert [SP07] who extracted LCS for a fixed start and end time using Marching Ridges [FP01]. They applied adaptive mesh refinement (AMR) to increase efficiency. Further adaptive methods also exist [GGTH07, HYX*20]. Lipinski and Mohseni [LM10] used the spatial and temporal coherence of FTLE to extract LCS for varying start times, restricting to the 2D space. Sadlo *et al.* [SRP11] extended this coherence-based idea to 3D. Recently, Stelter *et al.* [SWRT24] introduced a particle-based approach for series of integration times with a fixed start time.

Besides FTLE, there exist more properties and features based on the flow map. For instance, Lagrangian-averaged vorticity deviation (LAVD) integrates the normed difference between local and mean vorticity [HHFH16]. LAVD is particularly interesting as it observers vortex behavior over a time period in an objective way. For example, Abernathey and Haller [AH18] used LAVD to extract rotationally coherent Lagrangian vortices in the eastern Pacific. Another feature are recirculation surfaces which are 2-manifolds in the 5D space-time-time domain and encode positions where pathlines return to their exact start position after a finite duration. Wilde *et al.* [WRT18] searched for intersections on a regular grid to reconstruct and visualize these surfaces. Hofmann and Sadlo [HS19] used the dependent vectors operator for similar extractions. A recent approach is to circumvent the possibly error-prone reconstruction by direct visualization through ray tracing [SWT24]. Another flow map application are streak line fields [WT10] which enable an efficient computation of streak lines. Further, Wilde *et al.* [WRT20] proposed a method for deforming flow maps, similar to velocity field deformations [WTHS04, ZMT06, CKW*11]. These diverse applications highlight the importance of the flow map.

2.2. Computation, Approximation and Reconstruction

In this work we apply a decomposition and reconstruction of the flow map. Similar concepts exist in other fluid dynamics domains. Polthier and Preuß [PP03] applied Hodge decomposition to identify vector field singularities. This enables to describe the local geometric structure of instantaneous changes. Bhatia *et al.* [BNPB12] provided a thorough survey of such decompositions. Further, flow field clustering hierarchically models and reconstructs velocity data [GPR*04].

In contrast to Eulerian methods, we focus on the Lagrangian view. Due to the challenges with flow maps, multiple methods for approximation and reconstruction arose. Hummel *et al.* [HBJG16] estimated upper bounds of errors in pathlines reconstructed from short-time flow maps. Hlawatsch *et al.* [HSW10] proposed to accelerate flow map computation by parallel computation and concatenation of short-time flow maps. However, this comes with a tradeoff between accuracy and efficiency. Similar to [LM10], Brunton and Rowley [BR10] exploited flow map coherence for fast FTLE computation. Li *et al.* [LXS22] investigated pathline tracing using an interpolation of B-spline curves and their respective control points. Barakat and Tricoche [BT13] proposed an iterative method that employs a modified version of Sibson's interpolation. This captures geometric structures while reducing computational costs. However, their method is limited to fixed start and end times.

Large simulations are regularly carried out in a way that not every time-step is stored, because storing the data is more expensive than the actual simulation. In-situ friendly techniques directly process simulation data within the limited time windows [ACG*14, WRC*23]. Such methods produce highly accurate flow maps.

More recent work has shifted toward neural approaches. Jakob *et al.* [JGG20] trained a deep learning model to improve flow map interpolation. However, they only consider 2D divergence-free flows without obstacles or boundaries. Sahoo *et al.* [SLB22] introduced neural flow map reconstruction with high compression rates. However, their method only supports a fixed difference between start and end times. Further, it shows worse reconstruction results towards the temporal boundaries.

All techniques for flow map sampling and reconstruction mentioned so far have something in common: they do not consider the entire flow map but only a subset by setting the start time, the end time, or the difference between start time and end time to a constant value. Contrary, our approach is explicitly designed to cover the entire flow map.

3. Background

3.1. Velocity Fields and Flow Maps

We define a nD ($n = 2,3$) time-dependent velocity field $\mathbf{v}(\mathbf{x},t)$ as a vector valued function over the spatial domain $D \subseteq \mathbb{R}^n$ and a time interval $T = [t_s, t_e]$. We further consider the boundary ∂D of D that is a $(n-1)$-manifold. The velocity field maps each point in space-time to a spatial vector:

$$\mathbf{v} : D \times T \to \mathbb{R}^n , \quad (\mathbf{x},t) \to \mathbf{v}(\mathbf{x},t) . \tag{1}$$

Inside the domain of the velocity field we can seed a massless particle at the position $\mathbf{x}_0$ and time t_0 and compute its trajectory as an integral curve $\mathbf{c}(t)$. This is computed through numerical integration of the following ordinary differential equation:

$$\frac{d\mathbf{c}(t)}{dt} = \mathbf{v}(\mathbf{c}(t),t) , \quad \mathbf{c}(t_0) = \mathbf{x}_0 . \tag{2}$$

One can obtain positions on the particle's trajectory for a given destination time t_1. The period $t_1 - t_0$ is also called the integration time. The most typical way to encode the set of all particle trajectories inside a given domain is the flow map. It is a vector-valued function that describes the location of a particle seeded at $(\mathbf{x},t_0)$ and advected by the flow to the time t_1. Using the above definition (2), the flow map ϕ is defined as

$$\begin{aligned} \phi &: D \times T \times T \to D , \\ \phi(\mathbf{x},t_0,t_1) &= \mathbf{x} + \int_{t_0}^{t_1} \mathbf{v}(\mathbf{c}(t),t)dt . \end{aligned} \tag{3}$$

However, it is also possible that pathlines run out of domain, which is commonly the case for simulations. This creates „holes" in the domain where the result becomes undefined.

The flow map has two important properties:

$$\phi(\mathbf{x},t_0,t_0) = \mathbf{x} , \tag{4}$$

$$\phi(\mathbf{x},t_0,t_2) = \phi(\phi(\mathbf{x},t_0,t_1),t_1,t_2) . \tag{5}$$

Equation (4) denotes the flow map's identity, i.e., if start and end times are the same, a particle is mapped to its seeding position. Equation (5) is the additivity of the flow map, i.e., composing flow maps with a common time intermediate time t_1 maps to the same destination. This also means that a particle stays on the same pathline.

3.2. Finite-Time Lyapunov Exponent

The finite-time Lyapunov exponent (FTLE) [SLM05] is a scalar function which quantifies the rate of separation of infinitesimally close trajectories in a flow over a finite time interval. Large FTLE values indicate regions where particles diverge rapidly in the observation interval. It is directly related to Lagrangian Coherent Structures (LCS), which are material surfaces that delineate regions of distinct dynamical behavior in unsteady flows. These structures can be computed as the ridges of the respective FTLE field.

Given a velocity field $\mathbf{v}(\mathbf{x},t)$ and its associated flow map $\phi(\mathbf{x},t_0,t_1)$, the FTLE field is defined as follows:

$$\begin{aligned} &\mathrm{FTLE} : D \times T \times T \to \mathbb{R} , \\ &\mathrm{FTLE}(\mathbf{x},t_0,t_1) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{\max}(\nabla^\top \nabla)} \end{aligned} \tag{6}$$

where $\nabla := \nabla\phi(\mathbf{x},t_0,t_1)$ denotes the flow map gradient from t_0 to t_1, and $\lambda_{\max}$ is the largest eigenvalue of the right Cauchy-Green tensor $\nabla^\top \nabla$. The gradient is typically approximated using central differences. FTLE can be computed in forward ($t_0 < t_1$) and backward ($t_0 > t_1$) direction. The results correspond to repelling and attracting structures, respectively.

3.3. Lagrangian-Averaged Vorticity Deviation

The Lagrangian-averaged vorticity deviation (LAVD) is a measure designed to identify rotationally coherent structures in unsteady flows. Unlike FTLE, which focuses on the stretching of trajectories, LAVD measures the deviation of a particle's vorticity from the spatial mean vorticity along its trajectory. By considering the spatial mean, it provides an objective view of the data, i.e., the results remain unchanged under time-dependent spatial rotations and translations. Regions where the LAVD field exhibits tubular level surfaces correspond to rotationally coherent Lagrangian vortices. These are elements of the flow which show similar material rotation relative to the mean rotation under deformations of the fluid volume. Due to the objectivity and the focus on rotational coherence, LAVD is effective in identifying vortical structures in complex flows.

LAVD is defined for the 2D and the 3D space, we follow the descriptions of Haller *et al.* [HHFH16]. Given the velocity field $\mathbf{v}(\mathbf{x},t)$, the vorticity is $\omega(\mathbf{x},t) = \nabla \times \mathbf{v}(\mathbf{x},t)$, and the spatial mean vorticity $\bar{\omega}$ over the (possibly time-dependent) spatial domain $U(t) \subset \mathbb{R}^n$ is:

$$\bar{\omega}(t) = \frac{\int_{U(t)} \omega(\mathbf{x},t)\, dV}{\mathrm{vol}(U(t))} , \tag{7}$$

where $\mathrm{vol}(\cdot)$ is the volume for 3D flows and the area for 2D flows. Using this, the 3D variant of LAVD is defined as:

$$\begin{aligned} &\mathrm{LAVD} : D \times T \times T \to \mathbb{R} , \\ &\mathrm{LAVD}(\mathbf{x},t_0,t_1) = \int_{t_0}^{t_1} \|\omega(\phi(\mathbf{x},t_0,t),t) - \bar{\omega}(t)\|\, dt . \end{aligned} \tag{8}$$

For the definition in 2D space, the vorticity becomes a scalar function $\omega_3(\mathbf{x},t)$. It can be obtained from the 3D vorticity $\omega(\mathbf{x},t) = (0,0,\omega_3(\mathbf{x},t))^\top$. Consequently, the spatial mean vorticity is denoted as $\bar{\omega}_3(t)$. Therefore, the only changes are the two adapted functions ω_3 and $\bar{\omega}_3$, as well as using the absolute value $|\cdot|$ instead of the vector norm $\|\cdot\|$.

4. Methods

In this section we propose a framework to reduce the storage required for samplings of the flow map. We call this the *anchor flow map* (AFM) and will compare it to the naive sampling, referred to as the *full flow map* (FFM). The ideas are explained for 2D flows. Note that this works equivalently in 3D. In 2D, the flow map ϕ is a 4D map, i.e., a map from a 4D domain to 2D. In the following, we describe a method to represent this 4D map by three 3D maps. Afterwards, we present a concrete sampling and reconstruction strategy of our method. Finally, we show how to use our framework in combination with LAVD, highlighting the capacity of advancing our method with other methods related to the flow map.

4.1. Theoretical Foundation

In its core, our idea is to utilize the additivity property of the flow map, i.e., we compose two applications of the flow map. We start by selecting an anchor time $\tilde{t} \in T = [t_s, t_e]$. In theory, one can pick any $\tilde{t}$ within the time interval, but commonly we fix the value to

$$\tilde{t} = \tfrac{1}{2}(t_s + t_e)\ . \tag{9}$$

With this we define the first 3D map

$$\mathbf{d}_1 : D \times T \to D \times T\ ,$$
$$\mathbf{d}_1(\mathbf{x}, t_0) = \begin{cases} (\phi(\mathbf{x}, t_0, \tilde{t}), \tilde{t}) & \text{if the pathline stays in } D \\ (\phi(\mathbf{x}, t_0, t'), t') & \phi(\mathbf{x}, t_0, t') \in \partial D \text{ otherwise,} \end{cases} \tag{10}$$

with ∂D describing the boundary of the spatial flow domain D. It has the following interpretation: We start a particle integration from $(\mathbf{x}, t_0)$ towards the desired global anchor time $\tilde{t}$. If the integration reaches $\tilde{t}$ without leaving the domain D of start positions, then $\mathbf{d}_1(\mathbf{x}, t_0)$ encodes the destination of the successful integration. If the time $\tilde{t}$ is not reached because the particle flows out of the domain then $\mathbf{d}_1(\mathbf{x}, t_0)$ encodes the point on the domain boundary ∂D and the time t' where and when the particle leaves D. With this, the following two properties hold for any inputs $(\mathbf{x}, t_0) \in D \times T$ with $(\mathbf{x}', t') = \mathbf{d}_1(\mathbf{x}, t_0)$:

$$t' = \tilde{t} \quad \text{or} \quad \mathbf{x}' \in \partial D\ , \tag{11}$$
$$t' \in [\min(t_0, \tilde{t}), \max(t_0, \tilde{t})]\ . \tag{12}$$

The second property clarifies that t' is between t_0 and $\tilde{t}$, for both forward and backward integration. Note that $\mathbf{d}_1$ is a complete map that is uniquely defined for every $(\mathbf{x}, t_0) \in D \times T$.

The second 3D map

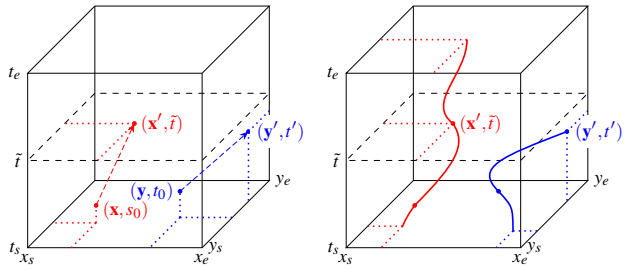
$$\mathbf{d}_2 : D \times T \to D\ , \quad \mathbf{d}_2(\mathbf{x}, t_1) = \phi(\mathbf{x}, \tilde{t}, t_1) \tag{13}$$

is a partial map: it may be undefined if the particle leaves the domain, but it is continuous in areas where it is defined. In contrast to $\mathbf{d}_1$, this function maps from the common anchor time $\tilde{t}$ to all end times $t_1 \in T$.

The third 3D map is defined as

$$\mathbf{d}_3 : \partial D \times T \times T \to D\ , \quad \mathbf{d}_3(\mathbf{x}, t', t_1) = \phi(\mathbf{x}, t', t_1)\ . \tag{14}$$

This function considers all pathlines for which the result of $\mathbf{d}_1$ has left the domain D. Like $\mathbf{d}_2$, it is a partial map. Note that for a 2D flow ∂D is a 1-manifold, and thus, $\mathbf{d}_3$ is indeed a 3D map.



(a) $\mathbf{d}_1(\mathbf{x}, s_0)$ *and* $\mathbf{d}_1(\mathbf{y}, t_0)$ **(b)** $\mathbf{d}_2(\mathbf{x}', \cdot)$ *and* $\mathbf{d}_3(\mathbf{y}', t', \cdot)$

Figure 2: *Two examples of the basic concept of the anchor flow map. Red: the integration of* $\mathbf{d}_1$ *stays inside the domain, so that the entire pathline is accessible from the position at the anchor time* $\tilde{t}$ *via* $\mathbf{d}_2$*. Blue: the integration of* $\mathbf{d}_1$ *ends at the domain boundary, so that the pathline is accessible from exit position* $\mathbf{y}'$ *and time* t' *by using* $\mathbf{d}_3$*.*

With this, the 4D flow map ϕ can be expressed by the three 3D maps $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ as

$$\phi(\mathbf{x}, t_0, t_1) = \begin{cases} \mathbf{d}_2(\mathbf{x}', t_1) & \text{if } \ t' = \tilde{t}, \\ \mathbf{d}_3(\mathbf{x}', t', t_1) & \text{otherwise,} \end{cases} \tag{15}$$
$$\text{with} \quad (\mathbf{x}', t') = \mathbf{d}_1(\mathbf{x}, t_0).$$

This describes the general approach of our anchor flow map: instead of sampling and reconstructing the full 4D flow map ϕ, we reduce this to a sampling of the 3D maps $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$. Figure 2 illustrates this idea for two pathlines, the red one requiring $\mathbf{d}_2$ and the blue one $\mathbf{d}_3$ for the reconstruction. For 3D flows, the definitions above hold as well. Here, we replace a 5D flow map ϕ by three 4D maps $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$.

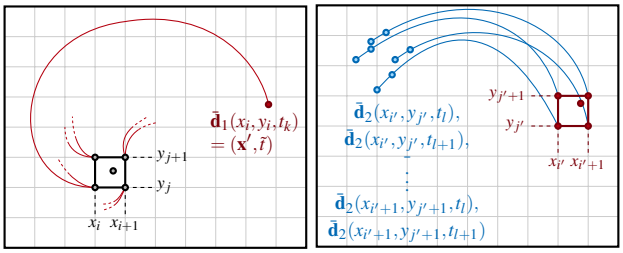
4.2. Sampling and Reconstruction

In the previous section, we explained the general framework for our approach. However, we still need to clarify how to actually sample and reconstruct the flow map. In this work, we apply multi-linear interpolation for all three maps $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$. For simplicity of the explanations, we restrict domains to hyper-rectangles, i.e., in 2D we utilize the domain $D = X \times Y$ with $X = [x_s, x_e]$ and $Y = [y_s, y_e]$ as well as $T = [t_s, t_e]$.

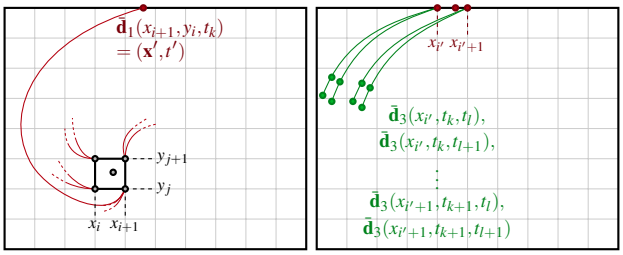
4.2.1. Redefinition of the Flow Map

To achieve a finite sampling, we first have to tackle a fundamental problem. Some flows are not only defined for a bounded domain but also for the entire space $\mathbb{R}^n$. An example is the well-known analytical ABC flow [DFG*86]: its spatial domain is normally chosen as $D = [0, 2\pi]^3$. However, one can still compute velocities and pathlines beyond these boundaries. Now consider our first map $\mathbf{d}_1$. As explained previously, our integration approach should stop at time $\tilde{t}$ or at the domain boundary ∂D. But what is ∂D here? Is it the boundary of the default domain, or does it even exist since we can compute vectors everywhere?

To decide this, we need to look at the map $\mathbf{d}_2$ with $D \times T \to D$. In context of sampling, the input must not be $\mathbb{R}^n$: we cannot sample an unbounded domain. But we cannot map to the restricted domain either, as this would make pathlines beyond the input domain impossible.



(a) $\bar{\mathbf{d}}_1$ *successfully integrates from* t_0 *to* $\tilde{t} \rightarrow$ *estimate via* $\bar{\mathbf{d}}_2$



(b) $\bar{\mathbf{d}}_1$ *stops integration at* $\partial\bar{D} \rightarrow$ *estimate via* $\bar{\mathbf{d}}_3$

Figure 3: *Illustration of our interpolation method for 2D space. (a) shows a case where the pathline of* $\mathbf{d}_1$ *stays inside* $\bar{D}$*, i.e., the reconstruction for that sample is performed via* $\mathbf{d}_2$*. In (b),* $\mathbf{d}_1$ *leaves* $\bar{D}$ *so that we use* $\mathbf{d}_3$ *in the second step. Note that two pathlines leave the same spatial point if we also interpolate the start time* t_0*. Similarly, we take two samples on the same pathline if* t_1 *is interpolated. In any case, eight samples are computed.*

Therefore, our solution is to adapt the definition of the flow map in Equation (3). The new definition differentiates between the domain of start positions $\bar{D}$ and the entire domain D for which the flow is defined:

$$\bar{\phi} : \bar{D} \times T \times T \rightarrow D \,, \tag{16}$$

with $\bar{D} \subseteq D$. The underlying velocity field $\mathbf{v}$ stays defined over the entire domain D, and the flow map still maps to D, too. However, now we consider the boundary $\partial\bar{D}$ of the bounded initial domain. In case of the ABC flow, we would choose $D = \mathbb{R}^n$ and $\bar{D} = [0, 2\pi]^3$. Consecutively, this leads to these new maps:

$$\begin{aligned} \bar{\mathbf{d}}_1 &: \bar{D} \times T \rightarrow \bar{D} \times T \,, \\ \bar{\mathbf{d}}_2 &: \bar{D} \times T \rightarrow D \,, \\ \bar{\mathbf{d}}_3 &: \partial\bar{D} \times T \times T \rightarrow D \,. \end{aligned} \tag{17}$$

Their respective definitions (see Section 4.1) are adapted analogously. The consequence is that pathlines running out of the initial domain $\bar{D}$ stops the integration for $\mathbf{d}_1$. In that case, one computes the flow map by using $\bar{\mathbf{d}}_3$. One can also apply this to simulation data, i.e., if one is interested in a subset of the possibly large domain, one can restrict the input. Of course, D and $\bar{D}$ can still coincide.

4.2.2. Sampling

Next, we show our actual sampling strategy for the three maps $\bar{\mathbf{d}}_1, \bar{\mathbf{d}}_2$ and $\bar{\mathbf{d}}_3$. In the following, fields marked with a hat ($\hat{}$) refer to discrete samplings of functions. First, the spatial and temporal dimensions are partitioned into a regular grid. Let $\hat{X}, \hat{Y}, \hat{T}$ denote the sampling cuts. The naive way to sample the flow map $\hat{\phi}$ is to compute and store it at each point of the grid $\hat{X} \times \hat{Y} \times \hat{T} \times \hat{T}$. In this case one computes the entire pathline once, starting at $\mathbf{x}_{(i,j)} = (x_i, y_j)$ and t_k to the temporal domain boundaries t_s and t_e (using forward and backward integration) and sample the destinations at each end time $t_l \in \hat{T}$. If a pathline leaves the domain, the respective samples are undefined. The reconstruction of $\hat{\phi}$ for the input $(\mathbf{x}, t_0, t_1)$ is obtained through quadri-linear interpolation.

For our approach, we also compute pathlines and sample them which is straightforward for $\bar{\mathbf{d}}_1$ and $\bar{\mathbf{d}}_2$: both functions use the grid $\hat{X} \times \hat{Y} \times \hat{T}$. For $\bar{\mathbf{d}}_1$, $\hat{T}$ refers to the start times t_0. Therefore, one must compute one pathline from all vertices (x, y, t_0) to $\tilde{t}$ and save (x', y', t'), i.e., the destination and end time of the integration. These results of the sampling are stored in $\hat{\mathbf{d}}_1$. For $\bar{\mathbf{d}}_2$, $\hat{T}$ represents the end time t_1, i.e., we compute the pathlines from (x, y) starting at the fixed time $\tilde{t}$, and apply backward integration to t_s as well as forward integration to t_e. These pathlines are sampled at all time steps $t_1 \in \hat{T}$ and stored in $\hat{\mathbf{d}}_2$.

For $\bar{\mathbf{d}}_3$, the sampling of pathlines requires a parametrization of the possibly free-form domain boundary ∂D. We define this reparametrization as:

$$\mathbf{p} : \partial D \rightarrow P \subset \mathbb{R}^{n-1} \,, \quad \mathbf{x} \rightarrow \mathbf{p}(\mathbf{x}) \tag{18}$$

which is a bijective map from all possible exit positions on ∂D to local coordinates P. Using the parametrization, one also has to select specific cut points for the sampling. In 2D, this is a 1D set which we denote as $\hat{P}$. Therefore, the respective 3D grid is defined as $\hat{P} \times \hat{T} \times \hat{T}$. To compute the pathlines, we reconstruct the 2D start positions from $\hat{P}$ using $\mathbf{p}^{-1}$ and compute the pathlines to t_s and t_e (if possible). Similarly to the other two maps, we save the results in $\hat{\mathbf{d}}_3$.

4.2.3. Reconstruction

Next, we consider the reconstruction of the flow map $\bar{\phi}$ for given input $(\mathbf{x}, t_0, t_1)$, as proposed in Equation (15). As a reminder: $\bar{\phi}$ is the flow map which differentiates between the input domain $\bar{D}$ and the global definition space D of the velocity field. Figure 3 illustrates the following concepts.

The reconstruction requires the composition of $\bar{\mathbf{d}}_1$ with either $\bar{\mathbf{d}}_2$ or $\bar{\mathbf{d}}_3$. First, we obtain the containing grid cell in $\hat{X} \times \hat{Y} \times \hat{T}$, consisting of eight vertices with indices $I = \{i, i+1\} \times \{j, j+1\} \times \{k, k+1\}$. For each vertex index we compute the respective weights $W = \{w_{(i,j,k)}, \ldots, w_{(i+1,j+1,k+1)}\}$ w.r.t. tri-linear interpolation. Next, for every vertex we look up the destination and intermediate time $(\mathbf{x}', t')$ in $\hat{\mathbf{d}}_1$. These data are further used in the second step: if $t' = \tilde{t}$ holds true we interpolate $\hat{\mathbf{d}}_2$ at $(\mathbf{x}', t_1)$, else we interpolate $\hat{\mathbf{d}}_3$ at $(\mathbf{p}(\mathbf{x}'), t', t_1)$. This is performed for all eight vertices. Each of the eight results is an approximation of the flow map at the respective vertex in I. Our reconstruction is the weighted average using the weights W.

It may happen that pathlines sampled in $\hat{\mathbf{d}}_2$ and $\hat{\mathbf{d}}_3$ left the global domain D. Consecutively, the respective samples will be undefined. If any of the results obtained from the second step is undefined, so is the result of the entire approximation.

4.3. Framework for Lagrangian Methods

For certain Lagrangian methods, one can use our method to additionally approximate and reconstruct the respective measurements. More specifically, the central requirement is that the measure builds on measurements along pathlines. This means that it must contain an integral which accumulates the data along the time. In this work we apply this to LAVD: indeed, it measures the vorticity along pathlines which makes it a valid candidate.

As for our main method, there are three maps. This time the target is the scalar LAVD value. Therefore, we adapt the three maps $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$ the following way:

$$\mathbf{l}_1 : D\times T \to D\times T\times\mathbb{R}\,, \quad \mathbf{l}_1(\mathbf{x},t_0) = (\mathbf{x}',t',lavd')\,, \tag{19}$$

$$l_2 : D\times T \to \mathbb{R}\,, \quad l_2(\mathbf{x},t_1) = \mathrm{LAVD}(\mathbf{x},\tilde{t},t_1)\,, \tag{20}$$

$$l_3 : \partial D\times T\times T \to \mathbb{R}\,, \quad l_2(\mathbf{x},t',t_1) = \mathrm{LAVD}(\mathbf{x},t',t_1)\,. \tag{21}$$

Equation (19) extends the definition of $\mathbf{d}_1$ (see Equation (10)) by the third component $lavd' = \mathrm{LAVD}(\mathbf{x},t_0,t')$. As for $\mathbf{d}_1$, if the pathline does not leave the domain, $t' = \tilde{t}$ holds true. Moreover, the map still encodes the destination position and time, together with the additional LAVD value. Note that l_2 and l_3, on the other hand, exclusively map to the LAVD value, and thus, become scalar functions.

Since the computation of LAVD (see Equation (8)) integrates the vorticity along pathlines, one can obtain the LAVD values and destinations simultaneously. Further, the definition contains the average vorticity $\bar{\omega}(t)$ which also depends on the time and can simply be observed. To compute LAVD from the three maps one must apply the additivity of the integral. Thus, we take the LAVD value from $\mathbf{l}_1$ and either add or subtract the value from l_2 or l_3. Using $(\mathbf{x}',t',lavd') = \mathbf{l}_1(\mathbf{x},t_0)$, we define it as:

$$\mathrm{LAVD}(\mathbf{x},t_0,t_1) = \begin{cases} \left| lavd' \odot \mathbf{l}_2(\mathbf{x}',t_1) \right| & \text{if } \; t' = \tilde{t}, \\ \left| lavd' \odot \mathbf{l}_3(\mathbf{x}',t',t_1) \right| & \text{otherwise.} \end{cases} \tag{22}$$

The operation $\odot$ adds or subtracts the second LAVD value depending on t_0, t_1 and t':

$$\odot = \begin{cases} + & \text{if } \; (t_0 \le t' \le t_1) \vee (t_1 \le t' \le t_0)\,, \\ - & \text{otherwise.} \end{cases} \tag{23}$$

For correct results in case of subtraction one must reapply the absolute value in Equation (22). Sampling and reconstruction can be applied as explained in Section 4.2.

4.4. Implementation

We implemented our methods in the Julia programming language (version 1.11.7) [BEKS17]. We share our code here: `https://visual2.cs.ovgu.de/daniel/afm.jl`. The additional material contains further algorithmic details.

For computing pathlines, we use an own implementation of a Runge-Kutta ODE solver of fourth order with adaptive step size control. It takes special care of domain boundaries, preventing out-of-domain integration and reliably detecting domain exits. The implementation of the full flow map is straightforward: we save the basic grid information (i.e., domain boundaries and resolutions) and an $(n+2)$-dimensional array. This array contains all nD destination vectors of the flow map for the grid vertices, i.e., all combinations of start positions, start times, and end times. By default we use `Float32` for memory efficiency.

For the anchor flow map, we propose an optimization of the theoretical definition. First, we split the map $\mathbf{d}_1$ into two two $(n+1)$-dimensional arrays: `d1_pos` which contains nD destination information, and `d1_bound` which saves one-byte boundary flags b. It encodes where the pathline integration ended:

- $b = 0$: the pathline stays inside D. `d1_pos` encodes the purely-spatial position, and the end time $\tilde{t}$ is known implicitly.
- $b \in \{1,\dots,2n\}$: the pathline ends at ∂D, and b encodes the respective side. Its constant spatial value is known implicitly, and inside `d1_pos` it is replaced by t'.

This way, we save n floats and one byte instead of $(n+1)$ floats. Additionally, the runtime decision whether to use $\mathbf{d}_2$ or $\mathbf{d}_3$ for reconstruction becomes an efficient byte comparison. $\mathbf{d}_2$ is saved as an array of nD vectors. $\mathbf{d}_3$ handles all $2n$ sides using separate arrays. The array of boundary side b is only computed if at least one pathline of $\mathbf{d}_1$ ended at that side, i.e., $b \in$ `d1_bound` holds true. This can reduce computation time and memory consumption.

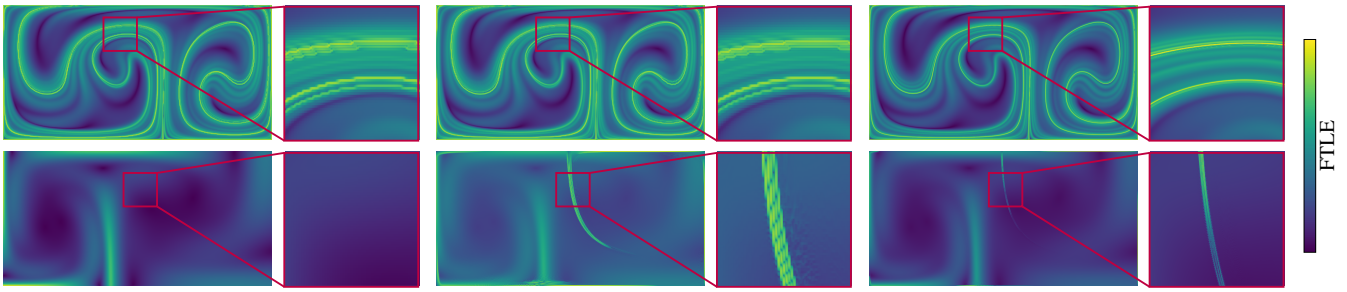
Next, we compare the memory cost for AFM and FFM. For simplicity we consider 2D flows and use 32 bit floats, but the discussion works for 3D and other precisions as well. In case of the FFM, the complexity is simple: we store 2D vectors (i.e., 8 bytes) on the space-time-time grid, resulting in $|\hat{X}|\cdot|\hat{Y}|\cdot|\hat{T}|\cdot|\hat{T}|\cdot 8$ bytes. For the AFM, both $\mathbf{d}_1$ and $\mathbf{d}_2$ build on the lower-dimensional grid $\hat{X}\times\hat{Y}\times\hat{T}$. Further, while $\mathbf{d}_2$ also maps to 8 bytes of vector data, $\mathbf{d}_1$ needs 9 bytes due to the `d1_bound` array. Thus, these two maps combined need $|\hat{X}|\cdot|\hat{Y}|\cdot|\hat{T}|\cdot 17$ bytes. Beyond that, $\mathbf{d}_3$ *might* add data, depending on the outflow along the domain boundaries. Assuming the extreme case of outflow for all boundaries, we require $2\cdot(|\hat{X}|+|\hat{Y}|)\cdot|\hat{T}|\cdot|\hat{T}|\cdot 8$ bytes. Although the formulas for the AFM look more complicated, each term has one factor less than for the FFM. For instance, consider an example where all dimensions have a resolution of 100. The FFM needs $100^4\cdot 8$ bytes = 800 MB. If the flow has no outflow, the AFM needs $100^3\cdot 17$ bytes = 17 MB. If all boundaries have outflow, this increases by $2\cdot 200\cdot 100^2\cdot 8$ bytes = 32 MB. This shows that we can significantly reduce the memory consumption. This effect mainly scales with the temporal resolution and the number of inflow/outflow sides.

The last point we mention considers the LAVD approximation. As described in Section 4.3, we need to save the LAVD *and* flow map values for $\mathbf{l}_1$. In contrast, for l_2 and l_3 it would suffice to only save the LAVD values. Nevertheless, we opted to also include the flow map data for the reconstruction which, in practice, allows for more flexible analysis of the flow. The main influence to the results of this paper is the respectively larger storage consumption.

5. Results

In this section we compare our anchor flow map (AFM) to the full flow map (FFM). All experiments were performed on a system with an Intel i7-13700 CPU with 24 threads and 64GB RAM, running on the Linux distribution EndeavourOS.

We consider different types of datasets. For 2D we use the Double



(a) *FFM, res_F* **(b)** *AFM$_c$, res_F* **(c)** *AFM$_c$, res_A*

Figure 4: *FTLE plots of the Double Gyre dataset for two examples. Top row: $t_0 = 1$, $t_1 = 19$. Here our method excels, as the far larger resolution leads to much more detailed FTLE computations. Bottom row: $t_0 = 16$, $t_1 = 19$. This shows the drawback of our method: even for the large resolution res_A, a thin but strong artifact arises.*

Dataset	T	Setup	Resolution $x \times y [\times z] \times t$	Method	Preprocess total / factor	Memory total / factor
Double Gyre	$[0, 20]$	res_F	$416 \times 208 \times 208$	FFM	34.1s / 1	29.9 GB / 1
				AFM$_c$	6.4s / 0.19	306 MB / 0.01
		res_A	$2416 \times 1208 \times 604$	AFM$_c$	650s / 19	30 GB / 1
Cylinder	$[0, 15]$	res_F	$588 \times 73 \times 295$	FFM	53.3s / 1	30 GB / 1
				AFM$_c$	18.3s / 0.34	314 MB / 0.01
		res_A	$3575 \times 446 \times 894$	AFM$_s$	3088s / 58	27.1 GB / 0.9
				AFM$_c$	1887s / 35	30 GB / 1
				AFM$_e$	2264s / 42	27.1 GB / 0.9
ABC	$[0, 10]$	res_F	$75 \times 75 \times 75 \times 75$	FFM	67.9s / 1	28.5 GB / 1
				AFM$_c$	16.8s / 0.25	3 GB / 0.11
		res_A	$132 \times 132 \times 132 \times 132$	AFM$_c$	132s / 1.9	29.4 GB / 1.03
Tangaroa	$[0, 2]$	res_F	$116 \times 69 \times 23 \times 116$	FFM	75.2s / 1	29.7 GB / 1
				AFM$_c$	20.6s / 0.27	1 GB / 0.03
		res_A	$320 \times 192 \times 64 \times 192$	AFM$_c$	700s / 9.3	29.7 GB / 1
Bickley Jet	$[0, 10]$	res_A	$2541 \times 762 \times 381$	AFM$_c$	2701s / –	30 GB / –

Table 1: *Overview of our experiments. We show preprocessing runtimes and memory consumptions, including factors relative to the FFM with res_F (lower is better). The subscript for the AFM method determines which anchor value $\tilde{t}$ is selected, with values t_s for AFM$_s$, t_e for AFM$_e$, and the center value for AFM$_c$.*

Gyre [SLM05] (analytical flow without any inflow or outflow) and the Cylinder flow [GGT17]. Latter one was simulated using the Gerris flow solver [Pop04] and has one inflow and one outflow side. For 3D, we apply the analytical ABC flow [DFG*86]. Originally, the dataset has periodic boundary conditions, but here we consider the flow map to map to $\mathbb{R}^3$. This gives a good example for the input domain restriction (Section 4.2.1) since all six boundaries have outflow. Further, we consider the simulated Tangaroa research vessel dataset [PSS04, Pop04], with one inflow and outflow side each. Lastly, for the evaluation of LAVD we apply our framework to the 2D Bickley Jet flow [RBBV*07]. As for the ABC flow, each boundary is passed by pathlines.

Additionally, we compare three different setups, all under the restriction that we limit the storage consumption to 30GB at maximum. Therefore, unless specified otherwise, we focus our comparison on the two different sampling resolutions res_F and res_A. They respectively define the maximum resolution so that either FFM or AFM does not exceed the 30GB limit. Note that for AFM this incorporates the implementation optimization that only relevant boundaries are computed for $\mathbf{d}_3$ (see Section 4.4). For an extensive comparison we consider the following three combinations:

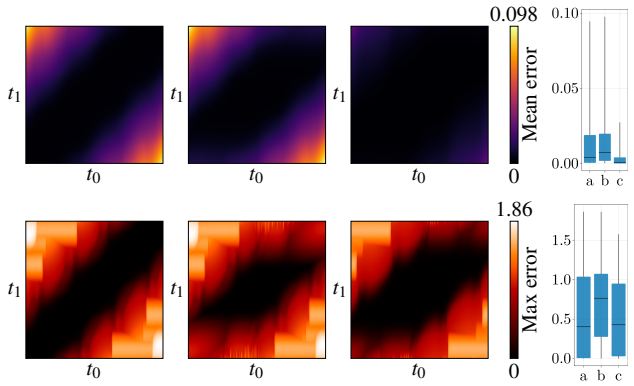
1. FFM with res_F (full flow map, max. 30GB),
2. AFM with res_F (our method, much less memory), and
3. AFM with res_A (our method, max. 30GB).

Therefore, we compare the FFM and the AFM using the same resolution, as well as using the same memory consumption. Commonly, we apply the standard choice of the anchor time stated in Equation (9). The only exception is the 2D Cylinder dataset. Here, we additionally apply and evaluate the cases $\tilde{t} \in \{t_s, t_e\}$, i.e., the anchor is the minimum or maximum time. Table 1 shows all combinations of setups examined in this work, as well as preprocessing times and memory consumptions. Note that due to inter-dimensional dependencies for an as-regular-as-possible grid it was not always possible to utilize the full range of the hard 30 GB limit. Further, for the Cylinder dataset with AFM$_s$ and AFM$_e$ the memory size is lower since for res_A we assumed the left and right bounds to have inflow and outflow. For these two, however, it was only one. Please also see the accompanying video and additional material for more results.

5.1. Accuracy

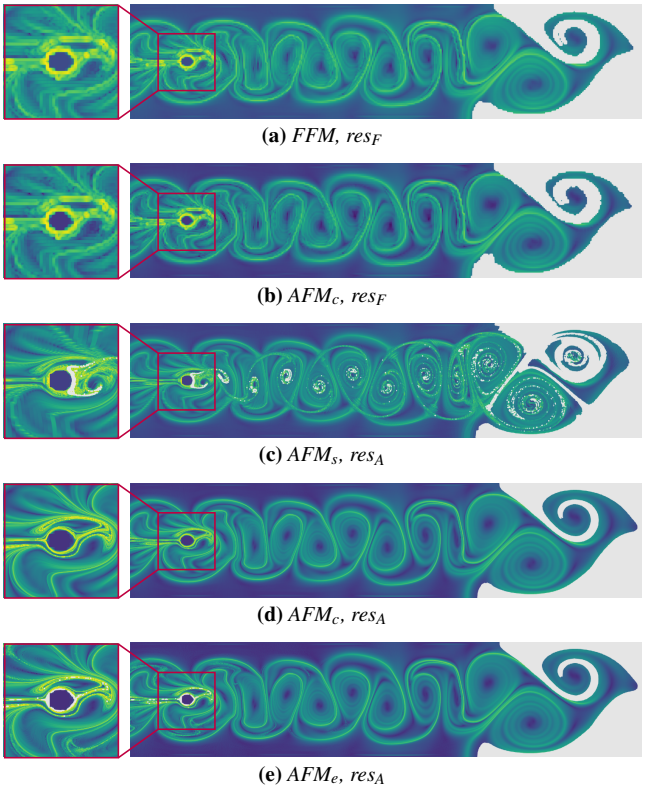
For evaluating the accuracy of the methods, we measure errors of the approximation methods. These errors are computed by taking the Euclidean norm of the approximation result and direct pathline integration which we consider as the ground truth. Given the space-time-time sampling grid for the approximators, we evaluate the errors in each cell center, i.e., so that all start positions, start times and end times have maximal distance to the samples at the cell vertices. We argue that this poses a fair study of expectable errors. Further, for the representation we summarize the results for each t_0–t_1 combination and save the mean and the maximum errors for each of these combinations.

Figure 5 shows t_0–t_1 plots of these errors for the Double Gyre dataset, together with boxplots of the statistical distribution. We refer to the additional material for plots of the other datasets and more in-depth information. First of all, it is unsurprising that approximation



(a) FFM, res_F (b) AFM$_c$, res_F (c) AFM$_c$, res_A

Figure 5: *Error plots for the Double Gyre dataset. The top row shows mean errors and the bottom row maximum errors for combinations of start and end times. To the right, boxplots provide extra information about the error distribution for the three setups. Blue boxes represent the interquartile range, also containing the median (black dash). The black vertical lines represent the total data range.*



(a) *FFM, res_F*

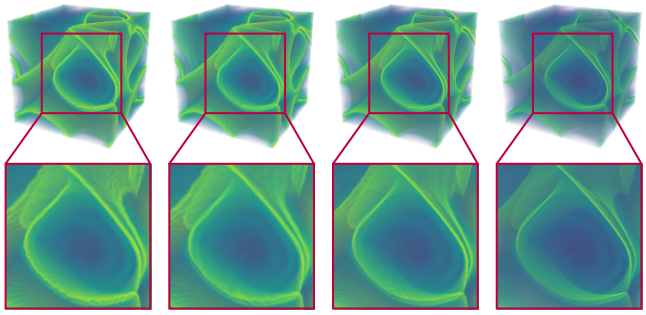
(b) *AFM$_c$, res_F*

(c) *AFM$_s$, res_A*

(d) *AFM$_c$, res_A*

(e) *AFM$_e$, res_A*

Figure 6: *FTLE plots of the Cylinder dataset for $t_0 = 6$, $t_1 = 10$. Light gray pixels encode positions where the flow map exited the domain, so that the FTLE cannot be computed. (a) and (b): The low-resolution approximators lead to similar results, where the values form blocky artifacts. (c) The AFM with $\tilde{t} = t_s$ results in strong, widespread artifacts, with wrong local maxima and failed FTLE computations. (d) and (e): Here, AFM achieves (very) good reconstructions, especially for the standard choice $\tilde{t} = t_c$.*



(a) *FFM, res_F* **(b)** *AFM$_c$, res_F* **(c)** *AFM$_c$, res_A* **(d)** *Integrated*

Figure 7: *Volume rendering of FTLE for the ABC flow with $t_0 = 2$, $t_1 = 8$. The results in (a) and (b) are still worse than in (c), but by a tinier margin than for the other datasets. Also, the ground truth, i.e., direct pathline integration in (d) looks much smoother.*

errors generally become larger for larger differences between t_0 and t_1, i.e., top left and bottom right corners. Regarding the mean errors, for the same resolution res_F FFM has a lower 25% quartile and median than AFM$_c$ which is better. However, AFM$_c$ with res_A achieves by far the best results with a median near zero and a much smaller error range. This shows that at same memory consumption we achieve much more precise approximations on average.

For the maximum errors, however, the advantage of AFM$_c$ with res_A compared to FFM becomes smaller. This comes due to the fact that our method must compute longer pathlines: if $t_0, t_1 < \tilde{t}$ or $t_0, t_1 > \tilde{t}$, the AFM first applies forward and then backward integration, or vice versa. Thus, the total integration length becomes larger, providing more potential for errors, especially in locations with large flow map gradients. This is also present in the maximum error plots of (b) and (c) which show increased values in the top right and bottom left corners. At the same positions, the FFM in (a) shows nearly zero errors.

Next, we consider visualizations computed using the flow map approximators through the application to FTLE. Coinciding with the improved mean errors, the AFM with the larger resolution res_A produces more detailed FTLE reconstructions than the FFM or AFM with res_F. For example, the top row in Figure 4 shows the FTLE field for the Double Gyre with a long integration time. Even for sharp features, AFM$_c$ with res_A creates precise ridges which is not possible for approximations with res_F: both the spatial and the temporal resolutions are too small, resulting in blocky structures and „double ridges". Latter come due to interpolations in the temporal dimensions where ridges move between the respective samples. This results in objectively false ridges. Therefore, only AFM$_c$ with res_A leads to plausible reconstructions. These more precise reconstructions can also be seen for the other datasets in Figures 1, 6 and 7. However, as discussed for the maximum error plots, our method induces new kinds of errors (see bottom row of Figure 4): for $t_0, t_1 < \tilde{t}$ or $t_0, t_1 > \tilde{t}$ (and especially for very short integration times), our method can show large approximation errors. This results in FTLE ridges in places where they should not occur at all. They come due to large flow map gradients from t_0 to $\tilde{t}$ and/or from $\tilde{t}$ to t_1.

Lastly, we compare the choice of the anchor time. Consider the

	Reconstruction		Pahtline integration			
	FFM	AFM	Double Gyre	Cylinder	ABC	Tangaroa
2D	0.41µs ± 0.23µs	1.6µs ± 0.39µs	6.5µs ± 4.2µs	25.7µs ± 29.6µs	–	–
3D	0.56µs ± 0.25µs	3.1µs ± 0.59µs	–	–	7.7µs ± 5µs	15µs ± 22.7µs

Table 2: *Runtime comparison of flow map evaluations with full flow map (FFM), anchor flow map (AFM), and direct pathline integration. Measurements show mean plus/minus standard deviation. Runtimes of FFM and AFM are measured for the Cylinder (2D) and Tangaroa (3D) datasets. The measurements for the other two datasets are nearly identical.*

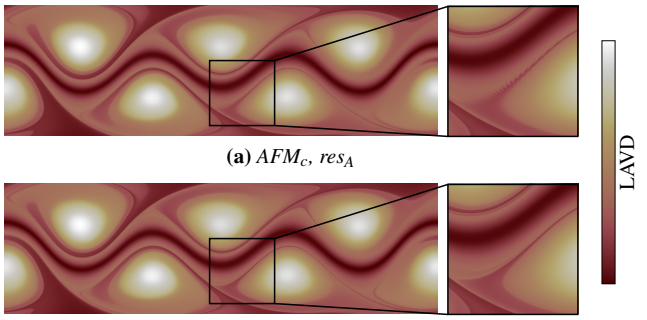
Cylinder dataset in Figure 6. As for the Double Gyre, FFM and AFM_c with the same resolution res_F provide comparable results. However, for res_A we applied three different anchor times: $t_s = 0$, $t_c = 7.5$, and $t_e = 15$. For the chosen time values $t_0 < t_c < t_1$ holds which is a comfortable configuration for AFM_c, and indeed, the result is convincing. Further, for AFM_e the result looks comparably good, but with minor artifacts of missing values. However, AFM_s creates a bad reconstruction with wrong FTLE ridges and many misses. This shows that strong turbulence on the way to the anchor time can pose a major problem. Additionally, we refer to the accompanying video which shows visualizations for varying times t_0, t_1. Unsurprisingly, one can clearly see that the closer $\tilde{t}$ is to t_0 and t_1, the better the visual appearance becomes. Therefore, our default choice in Equation (9) is reasonable if one is interested in the whole time domain: it minimizes the mean difference between $\tilde{t}$ and t_0, t_1. If a specific region of times is of interest, changing $\tilde{t}$ to a time in that region is advised.

5.2. Runtime

In terms of runtime, we consider the preprocessing time and the actual evaluation time (i.e., performing the reconstruction). While the preprocessing time may be very long for both methods, it only has to be performed once. Afterwards, all kinds of methods which rely on the flow map can make advantage of the fast reconstruction. This especially makes reconstruction methods interesting for interactive applications, or if the flow map is required for multiple methods. As part of our work, we implemented such an interactive application with real-time computations of FTLE and LAVD fields.

Table 1 shows the preprocessing times for all datasets and methods. These results are unsurprising: with the same resolution res_F, the AFM needed just between 16% and 34% of the time of the FFM since it only computes a subset of the pathlines. For res_A, however, AFM_c has factors between 1.9 (ABC) and 35 (Cylinder) due to the larger resolutions, and thus, significantly more pathline integration. Further, for the 2D Cylinder we compared AFM_c to AFM_s and AFM_e. Choosing $\tilde{t}$ as the center time results in a shorter preprocess than for t_s and t_e with factors 58 and 42. The reason: the sum of pathline lengths become substantially longer, and our ODE solver might need to handle more domain exits.

In contrast to the preprocessing, the reconstruction times are reasonably stable. The only obvious changes come due to 2D versus 3D. However, runtimes of pathline integrations depend on the datasets. Therefore, we compare following methods: the four combinations of



(a) AFM_c, res_A

(b) *Integrated*

Figure 8: *LAVD plots of the Bickley Jet dataset for $t_0 = 0$, $t_1 = 10$. As visible in the closeup, partly there arise small artifacts on the reconstruction, but in general our method works very well.*

FFM and AFM with 2D and 3D, and pathline integration with four datasets. For the evaluation, we compute one million flow map samples with randomized start positions, start times and end times. The results can be found in Table 2. As expected, the reconstruction with FFM is by far the fastest method – in terms of fast computation, no method can compete with the simplicity of multilinear interpolation. But still, with respect to the mean runtimes the AFM is between 2.5 and 16 times faster, with larges differences for simulation datasets (Cylinder and Tangaroa). Also note the large standard deviations for integration. This comes due to much shorter runtimes for small differences of start and end times. Long integration times, however, take much longer. The reconstruction methods, on the other hand, are agnostic to the start and end times. Beyond that, we applied multi-linear interpolation for the simulated flow data. If one would apply C1 continuous reconstructions or unstructured meshes, their runtimes further increase.

5.3. Application to LAVD

In Section 4.3 we showed how to use our method as a framework for Lagrangian methods. We applied this framework to the computation of LAVD fields for the Bickley Jet flow [RBBV*07]. Figure 8 compares the visual result of our method to direct pathline integration. The start and end times in this plot coincide to the bounds of our time interval T, i.e., it shows the result for the maximal integration time. As for the results of FTLE, the accompanying video shows a more extensive comparison with varying combinations.

The results are nearly identical to direct pathline integration for the largest part of the domain, which is a good result. However, as one can see in the closeup of Figure 8, in few places small artifacts arise. Thus, as for the reconstruction of the flow map, our method induces some errors to LAVD. However, these artifacts do only occur in few, small regions. Further, the maxima of LAVD (which are the most interesting properties) seem to be reconstructed reliably. Therefore, we see this as a satisfactory result.

6. Discussion and Limitations

First method to sample the entire flow map. To the best of our knowledge, our approach of anchor flow maps is the first one to tackle the

sampling of the entire flow map, i.e., covering the entire space-time-time domain. Contrary, other existing approaches do only consider subsets of this domain (e.g., fixed start times, fixed integration times). Extending them to the entire flow map (for instance by naively copying them for each start and/or end time) applies them in a domain they were not developed for, resulting in an undocumented behavior. Because of this, we restrict ourselves to a comparison to a regular sampling (the full flow map) as base-line.

Improving resolutions. As shown in Table 1, our method enables to compute and store much larger resolutions of the flow map as it reduces the memory cost by one order of magnitude. This is the biggest advantage compared to the common full flow map sampling. The magnitude of the difference also depends on the inflow and outflow characteristics of the underlying flow. The two extreme examples are the Double Gyre (neither inflow nor outflow) and the ABC flow (inflow and outflow through all six sides): For the Double Gyre, we were able to multiply the spatial resolutions by a factor of around 6 and the temporal resolutions by 3 while maintaining the memory consumption. For the ABC flow, however, it was less than a factor of 2 for all dimensions. Still, even this increase leads to better precisions and more detailed feature extractions.

Conservative interpolation. Flow maps are partial maps that may be undefined. This occurs when a trajectory leaves the spatial domain of the flow. Sampling-based approaches give even larger areas of undefined flow maps since interpolation is done in a conservative way: an interpolated flow map becomes undefined if at least one involved sample is undefined. Again, considering same resolutions for both our and the full flow map approximations, the areas of „false undefined" values are larger than in a sampling of the full flow map because of the concatenation of two interpolations. However, this is drastically improved when we apply larger resolutions for the AFM than for the FFM (see the table in the additional material).

Choice of the anchor time. The choice of the anchor time $\tilde{t}$ has influence on our approach. It gives good reconstructions for start and end times t_0, t_1 that are on „opposite" sites of $\tilde{t}$, i.e., $t_0 \leq \tilde{t} \leq t_1$ and $t_1 \leq \tilde{t} \leq t_0$, but tends to give larger errors for $\tilde{t} < t_0, t_1$ and $\tilde{t} > t_0, t_1$ (see Section 5.1). Since we consider large integration times (i.e., large $\|t_1 - t_0\|$) to be the more interesting part of flow maps in most applications, the choice (9) is reasonable: It minimizes the domain in which $\tilde{t} < t_0, t_1$ and $\tilde{t} > t_0, t_1$ holds true and gives good coverage for long integration times. If other smaller time differences are of interest, $\tilde{t}$ can be moved to another value between t_s and t_e. The influence can be seen in our comparison for the Cylinder dataset. This, however, makes a re-computation of the AFM necessary.

Performance of pre-computations. Clearly, the pre-computations of flow maps are computationally expensive (although it can easily make use of parallelization). This also holds for pre-computations of our anchor flow maps. We note that the computation of $\mathbf{d}_1$ is significantly more expensive than for $\mathbf{d}_2$ and $\mathbf{d}_3$ because for $\mathbf{d}_1$ a new path line integration is requested for each sample point. Contrary, multiple sample points are covered by one path line for $\mathbf{d}_2$ and $\mathbf{d}_3$. Moreover, $\mathbf{d}_2$ might be a partial map so that some pathlines might not be integrated to the bounds of T, and for $\mathbf{d}_3$, we can even skip computations of parts of the domain boundary. Nonetheless, for same grid resolutions our method requires significantly less pathline integration than the full flow map. In detail, we only have to compute a subset of the pathlines present in the full flow map. Thus, this reduces the computational effort by a fair amount (see Table 1). Obviously, applying larger resolutions still causes much longer pre-computation times.

Restriction to regular sampling. In this paper, we restricted ourselves to a regular sampling of both the FFM and our lower-dimensional AFM. For none of them, we considered adaptive sampling techniques. We believe that this is a fair comparison because of the following reason: adaptive sampling is an approach orthogonal to our approach. While a better accuracy of the FFM can be expected by adaptive samplings, the same improvement can be expected by an adaptive sampling of the fields $\mathbf{d}_1, \mathbf{d}_2$ and $\mathbf{d}_3$. The main purpose of the paper is to study the impact of the lower-dimensional representation of flow maps. Even with the "naive" approach of regular sampling, the AFM was able to give real-time access to the entire flow map. We are not aware of any other techniques which provide this. Clearly, further accuracy improvements can be expected by combining our approach with adaptive sampling techniques. However, this is not a trivial advantage: adaptive structures like quadtrees or unstructured triangle meshes need more time to locate the cell of interest. Simultaneously, triangles and tetrahedra have fewer vertices than (hyper-)rectangles, decreasing evaluation workload. Exploring the accuracy and runtime behavior of such methods can be an exciting topic for future work.

Handling arbitrarily shaped domains. The sampling in this paper was exclusively described for (hyper-)rectangular domains. In principle, the theory of anchor flow maps as proposed in Section 4.1 is capable of handling complex domains and boundaries, e.g., by using unstructured grids. The crucial part is to explicitly define the domain boundary ∂D which is part of the definition of $\mathbf{d}_3$. Moreover, it is important to describe all sections of ∂D that have inflow or outflow. This enables an efficient restriction to relevant sections of $\mathbf{d}_3$. For regular domains with uniform grids both parts are comparably simple (see Section 4.4). However, for unstructured grids this may need a complicated boundary parametrization (see Equation 18) and inflow and outflow detection. Further, a current limitation is that we do not consider flows with periodic boundary conditions. This might require specific considerations for theory and implementation.

7. Conclusions

In this paper we presented the anchor flow map, a new method for the representation of flow maps. It is driven by replacing the original $(n+2)$-dimensional flow map by three lower-dimensional functions. We used this technique to sample and reconstruct the flow map for its entire space-time-time domain. This new approach enables to either reduce the memory consumption significantly, or to apply much larger resolutions. Latter led to much more accurate reconstructions compared to native sampling of the full flow map. When applying the anchor flow map with the larger resolutions we were able to compute more detailed approximations of FTLE fields. Further, we showed how to expand our method for the computation of LAVD. There are various directions for future work, e.g., by combining our framework with other, orthogonal approximation methods for the flow map (see Section 6). Additionally, one can search for other features to integrate to our method, as we did with LAVD.

References

[ACG*14] AGRANOVSKY A., CAMP D., GARTH C., BETHEL E. W., JOY K. I., CHILDS H.: Improved post hoc flow analysis via lagrangian representations. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)* (2014), IEEE, pp. 67–75. doi:10.1109/LDAV.2014.7013206. 2

[AH18] ABERNATHEY R., HALLER G.: Transport by lagrangian vortices in the eastern pacific. *Journal of Physical Oceanography 48*, 3 (2018), 667–685. doi:10.1175/JPO-D-17-0102.1. 2

[BEKS17] BEZANSON J., EDELMAN A., KARPINSKI S., SHAH V. B.: Julia: A fresh approach to numerical computing. *SIAM Review 59*, 1 (2017), 65–98. doi:10.1137/141000671. 6

[BNPB12] BHATIA H., NORGARD G., PASCUCCI V., BREMER P.-T.: The helmholtz-hodge decomposition—a survey. *IEEE Transactions on visualization and computer graphics 19*, 8 (2012), 1386–1404. doi:10.1109/TVCG.2012.316. 2

[BR10] BRUNTON S. L., ROWLEY C. W.: Fast computation of finite-time lyapunov exponent fields for unsteady flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science 20*, 1 (2010). doi:doi.org/10.1063/1.3270044. 2

[BT13] BARAKAT S. S., TRICOCHE X.: Adaptive refinement of the flow map using sparse samples. *IEEE transactions on visualization and computer graphics 19*, 12 (2013), 2753–2762. doi:10.1109/VIS54862.2022.00037. 2

[CKW*11] CHEN G., KWATRA V., WEI L.-Y., HANSEN C. D., ZHANG E.: Design of 2d time-varying vector fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 10 (2011), 1717–1730. doi:10.1109/TVCG.2011.290. 2

[DFG*86] DOMBRE T., FRISCH U., GREENE J. M., HÉNON M., MEHR A., SOWARD A. M.: Chaotic streamlines in the abc flows. *Journal of Fluid Mechanics 167* (1986), 353–391. doi:10.1017/S0022112086002859. 4, 7

[FP01] FURST J. D., PIZER S. M.: Marching ridges. In *SIP* (2001), vol. 1, Citeseer, pp. 22–26. 2

[GGT17] GÜNTHER T., GROSS M., THEISEL H.: Generic objective vortices for flow visualization. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (2017), 1–11. doi:10.1145/3072959.3073684. 7

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HANS H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1464–1471. doi:10.1109/TVCG.2007.70551. 2

[GPR*04] GRIEBEL M., PREUSSER T., RUMPF M., SCHWEITZER M. A., TELEA A.: Flow field clustering via algebraic multigrid. In *IEEE Visualization 2004* (2004), IEEE, pp. 35–42. doi:10.1109/VISUAL.2004.32. 2

[Hal01] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena 149*, 4 (2001), 248–277. doi:10.1016/S0167-2789(00)00199-8. 2

[HBJG16] HUMMEL M., BUJACK R., JOY K. I., GARTH C.: Error estimates for lagrangian flow field representations. In *EuroVis (Short Papers)* (2016), pp. 7–11. doi:10.2312/eurovisshort.20161153. 2

[HHFH16] HALLER G., HADJIGHASEM A., FARAZMAND M., HUHN F.: Defining coherent vortices objectively from the vorticity. *Journal of Fluid Mechanics 795* (2016), 136–173. doi:10.1017/jfm.2016.151. 2, 3

[HS19] HOFMANN L., SADLO F.: The dependent vectors operator. *Computer Graphics Forum 38*, 3 (2019), 261–272. doi:10.1111/cgf.13687. 2

[HSW10] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE transactions on visualization and computer graphics 17*, 8 (2010), 1148–1163. doi:10.1109/TVCG.2010.227. 2

[HY00] HALLER G., YUAN G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena 147*, 3-4 (2000), 352–370. doi:10.1016/S0167-2789(00)00142-1. 2

[HYX*20] HANG H., YU B., XIANG Y., ZHANG B., LIU H.: An objective-adaptive refinement criterion based on modified ridge extraction method for finite-time lyapunov exponent (ftle) calculation. *Journal of Visualization 23* (2020), 81–95. doi:10.1007/s12650-019-00605-1. 2

[JGG20] JAKOB J., GROSS M., GÜNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2020), 1279–1289. doi:10.1109/TVCG.2020.3028947. 3

[LM10] LIPINSKI D., MOHSENI K.: A ridge tracking algorithm and error estimate for efficient computation of lagrangian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science 20*, 1 (2010). doi:10.1063/1.3270049. 2

[LXS22] LI H., XIONG T., SHEN H.-W.: Efficient interpolation-based pathline tracing with b-spline curves in particle dataset. In *2022 IEEE Visualization and Visual Analytics (VIS)* (2022), IEEE, pp. 140–144. doi:10.1109/TVCG.2013.128. 2

[Pop04] POPINET S.: Free computational fluid dynamics. *ClusterWorld 2*, 6 (2004). URL: http://gfs.sf.net/. 7

[PP03] POLTHIER K., PREUSS E.: Identifying vector field singularities using a discrete hodge decomposition. In *Visualization and mathematics III*. Springer, 2003, pp. 113–134. doi:10.1007/978-3-662-05105-4_6. 2

[PSS04] POPINET S., SMITH M., STEVENS C.: Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. *Journal of Atmospheric and Oceanic Technology 21*, 10 (2004), 1575–1589. doi:10.1175/1520-0426(2004)021<1575:EANSOT>2.0.CO;2. 7

[RBBV*07] RYPINA I., BROWN M. G., BERON-VERA F. J., KOÇAK H., OLASCOAGA M. J., UDOVYDCHENKOV I.: On the lagrangian dynamics of atmospheric zonal jets and the permeability of the stratospheric polar vortex. *Journal of the Atmospheric Sciences 64*, 10 (2007), 3595–3610. doi:10.1175/JAS4036.1. 7, 9

[SLB22] SAHOO S., LU Y., BERGER M.: Neural flow map reconstruction. *Computer Graphics Forum 41*, 3 (2022), 391–402. doi:10.1111/cgf.14549. 3

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3-4 (2005), 271–304. doi:10.1016/j.physd.2005.10.007. 2, 3, 7

[SP07] SADLO F., PEIKERT R.: Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE transactions on visualization and computer graphics 13*, 6 (2007), 1456–1463. doi:10.1109/TVCG.2007.70554. 2

[SRP11] SADLO F., RIGAZZI A., PEIKERT R.: Time-dependent visualization of lagrangian coherent structures by grid advection. *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications* (2011), 151–165. doi:10.1007/978-3-642-15014-2_13. 2

[SWRT24] STELTER D., WILDE T., RÖSSL C., THEISEL H.: A particle-based approach to extract dynamic 3d ftle ridge geometry. In *Computer Graphics Forum* (2024), Wiley Online Library, p. e15203. doi:10.1111/cgf.15203. 2

[SWT24] STELTER D., WILDE T., THEISEL H.: Ray tracing for recirculation surfaces. In *Vision, Modeling, and Visualization* (2024), Linsen L., Thies J., (Eds.), The Eurographics Association. doi:10.2312/vmv.20241207. 2

[WRC*23] WOLLIGANDT S., RÖSSL C., CHI C., THÉVENIN D.,

THEISEL H.: Autonomous particles for in-situ-friendly flow map sampling. In *VMV* (2023), pp. 189–197. doi:10.2312/vmv.20231242. 2

[WRT18] WILDE T., RÖSSL C., THEISEL H.: Recirculation surfaces for flow visualization. *IEEE transactions on visualization and computer graphics 25*, 1 (2018), 946–955. doi:10.1109/TVCG.2018.2864813. 2

[WRT20] WILDE T., RÖSSL C., THEISEL H.: Flow map processing by space-time deformation. In *Advances in Visual Computing* (2020), Springer, pp. 236–247. doi:10.1007/978-3-030-64556-4_19. 2

[WT10] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1225–1234. doi:10.1109/TVCG.2010.198. 2

[WTHS04] WEINKAUF T., THEISEL H., HEGE H.-C., SEIDEL H.-P.: Topological construction and visualization of higher order 3d vector fields. *Computer Graphics Forum 23*, 3 (2004), 469–478. doi:10.1111/j.1467-8659.2004.00778.x. 2

[ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Transactions on Graphics (ToG) 25*, 4 (2006), 1294–1326. doi:10.1145/1183287.1183290. 2