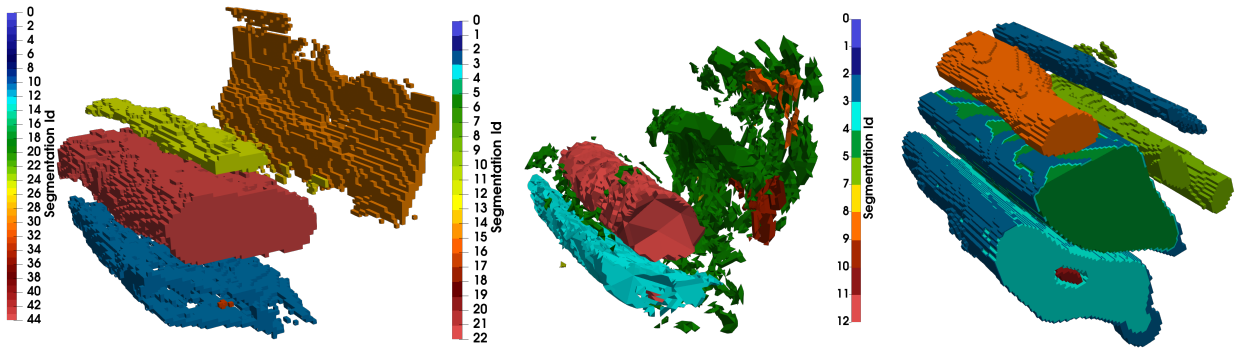
The Effect of Data Transformations on Scalar Field Topological Analysis of High-Order FEM Solutions

Ashok Jallepalli, Joshua A. Levine, and Robert M. Kirby



(a) Sampled vorticity. (b) Subdivided vorticity. (c) L-SIAC vorticity.

Fig. 1: Topological segmentation of counter-rotating vortex sampled using different methodologies discussed in the paper and by filtering the contour tree for segments that resemble vortex-like structures. The number, shape, and boundaries of the segments are different for the three techniques.

Abstract— High-order finite element methods (HO-FEM) are gaining popularity in the simulation community due to their success in solving complex flow dynamics. There is an increasing need to analyze the data produced as output by these simulations. Simultaneously, topological analysis tools are emerging as powerful methods for investigating simulation data. However, most of the current approaches to topological analysis have had limited application to HO-FEM simulation data for two reasons. First, the current topological tools are designed for linear data (polynomial degree one), but the polynomial degree of the data output by these simulations is typically higher (routinely up to polynomial degree six). Second, the simulation data and derived quantities of the simulation data have discontinuities at element boundaries, and these discontinuities do not match the input requirements for the topological tools. One solution to both issues is to transform the high-order data to achieve low-order, continuous inputs for topological analysis. Nevertheless, there has been little work evaluating the possible transformation choices and their downstream effect on the topological analysis. We perform an empirical study to evaluate two commonly used data transformation methodologies along with the recently introduced L-SIAC filter for processing high-order simulation data. Our results show diverse behaviors are possible. We offer some guidance about how best to consider a pipeline of topological analysis of HO-FEM simulations with the currently available implementations of topological analysis.

Index Terms—High-Order Finite Element Methods, Filtering Techniques, Scalar Field Visualization, Topological Analysis.

1 Introduction

The development of robust, efficient solvers that utilize high-order finite element methods (HO-FEM), also sometimes classified as spectral/*hp* element methods, is an area of considerable interest to the simulation community at present. The use of higher order polynomial expansions within elements carries a number of benefits, as seen from two main perspectives. 1) Numerically, these methods exhibit far lower levels of numerical dispersion and dissipation at higher polynomial orders. This makes them a particularly well-suited approximation choice in areas such as computational fluid dynamics, where the accurate time-advection of energetic structures such as vortices is a key concern (e.g., [36, 44]). 2) Given current hardware trends, the most appealing property of these methods in recent years has been their computational performance. Although the cost per degree of freedom in terms of algorithmic floating point operations (FLOPS) increases substantially with polynomial order, the use of higher order expansions leads to formulations of the underlying equations of state that involve dense, compact kernels for key finite element operators, such as inner products and derivatives. This is important from the perspective of modern hardware, where increasingly the bottleneck in performance is memory bandwidth as opposed to the clock speed of processors. The underlying arithmetic intensity of the algorithm at hand (i.e., the number of floating-point operations performed for each memory operation) is therefore key to attaining optimal performance. This is where high-order methods have a significant advantage over lower order methods (e.g., [27, 50]).

These trends indicate that there will continue to be an increasing need for visualization of high-order finite element solutions. Topological analysis has a rich history of providing methods to extract and visualize structural properties in simulation data. For example, one can use topology to study vortex breakdown patterns [69], vortex merging [3], and shedding patterns. These techniques are built with the assumption of continuity in the data, but as presented in [37], one of the challenges associated with the visualization, and in this case the topological analysis, of HO-FEM fields (and their corresponding derived fields) is their lack of continuity at element interfaces. Fundamentally, topological analysis is based on extracting properties in a continuum: these properties are designed to be invariant to deformation but sensitive to topological events such as cutting or splitting. At first glance, it seems an impossible task to analyze a discontinuous function using standard topological properties, but we note that even if the solution is represented with a discontinuous approximation, the simulated

- *Ashok Jallepalli and Robert M. Kirby are with SCI Institute, University of Utah. E-mail: {ashokj,kirby}@sci.utah.edu.*
- *Joshua A. Levine is with Department of Computer Science, University of Arizona. E-mail: josh@email.arizona.edu.*

phenomena of interest are often assumed to be continuous (from the physical modeling perspective).

Besides the issues with discontinuous data, a practical problem is that most of the mathematics and implementations of topological analysis for scalar fields focus on piecewise linear interpolants. There has been limited work in pushing the limits beyond piecewise linear interpolants, and the typical extensions consider only a specific interpolant [1, 52, 55] as opposed to general methodologies for high-order interpolants [9, 53]. Chen et al. considered the effects of interpolants on Morse decompositions of vector fields, but did not explicitly use the high-order, discontinuous methods we target [13]. This lack of prior art is justifiable, as piecewise linear representations have the advantage that they help to limit the space of combinatorial possibilities within a given mesh (e.g., in a piecewise linear element, there may be only one critical point). The typical assumption is that any more complex functions can be represented by simply increasing mesh resolution. Nevertheless, using increased mesh resolutions can require an infeasible amount of storage overhead, and as we show in our work, they may not always achieve an ideal characterization of the shape of topological features.

To bridge the gap between HO-FEM simulations and topological methods, we need to either transform the data or redesign (and re-verify) the topological techniques. In this work, we consider the first option. Specifically, we explore three main methodologies for converting the HO-FEM data to match the input requirements of topological tools. We then conduct an empirical study to analyze the effect of these methodologies on downstream topological analyses. Two of the methodologies we consider are sampling the data on a regular grid (avoiding discontinuities) and subdividing the elements on the simulation mesh (avoiding discontinuities and judiciously producing low-order elements). These methodologies are commonly used due to their ease of implementation and lower computational cost, but they have the disadvantage of creating undesirable artifacts [37]. We use these methodologies to provide baselines to discuss the interplay between the sampling parameter and its effect on the intensity of the discontinuities and interpolation artifacts. We also provide insight into selecting the sampling parameter to help reduce the artifacts. Simultaneously, we consider using topological analysis to filter out the artifacts as well as identifying what significant features may be retained. The third methodology we consider is the Line-SIAC (L-SIAC) filter, which is computationally expensive, but has been successful in identifying a more extended range of features compared to the other two techniques.

In this paper, we demonstrate that through the use of the L-SIAC filtering methodology, we can transform HO-FEM data so as to take full advantage of the suite of techniques and intuitions developed within the topological analysis field. Furthermore, we provide a set of insights into how these tools – L-SIAC and topological analysis – interact, which helps show how they might be used together in the future.

Specifically, our contributions are

- An empirical study to evaluate data transformations of high-order scalar fields that enable the use of existing topological analysis tools;
- Experimentation using three different data transformation methods and two different datasets; and
- Evaluation of the downstream effects in topological structures including persistence diagrams and curves, the position and number of critical points, and the segmentation based on the contour tree.

These contributions are the first step towards a better understanding of existing issues between data interpolant and topological feature extraction. Such a study may help motivate the development of new topological techniques that can adapt to HO-FEM data without the need to first transform it.

2 Previous Work

In this section, we review the main concepts of topological analysis in regard to scalar fields for linear and high-order polynomial data. The current challenges are to extend these techniques to HO-FEM data and to introduce the data transformation methodologies. These methodologies transform the data into continuous piecewise linear data to enable topological analysis.

2.1 Topological Analysis

We review the main concepts and structures of topological analysis that we employ in our empirical study. For a more complete introduction to the mathematics of computational topology, see Edelsbrunner and Harer [24]. For details of their usage, Heine et al. provide a recent survey of topology-based methods in visualization and analysis [35].

Topological analysis provides a collection of tools to extract features from data that characterize its structure. Moreover, it also provides mechanisms to rank and filter these features, thus offering an analyst the ability to summarize a scalar field at multiple scales. While this general framework is applicable to a variety of data modalities, a key area of focus in the visualization community is the analysis of scalar fields. From scalar field data, one can visualize the structure through collections of important feature points (i.e., critical points [4]), graph structures that encode level sets (i.e., Reeb graphs [5, 56, 67] and contour trees [10]), and segmentations of gradient flow behavior (i.e., Morse-Smale complexes [19, 32]). The structures have seen direct usage in visualization of scalar fields, for example in selecting isosurfaces [70], topologically-guided simplification [68], feature tracking [61], transfer function design [73], isosurface simplification [11], and similarity estimation [65]. Even though topological analysis is a purely mathematical framework, a key reason for its success has been the mapping of these mathematical abstractions to application-specific features, as demonstrated by its use across a wide variety of domains, including astrophysics [59, 62], battery design [33], combustion [7, 18, 31], chemistry [12, 29], porous media [34], turbulence [42], and vortex extraction [41].

In this paper, we focus on level set topology as our main tool of interest. Let $f : \mathscr{M} \to \mathbb{R}$ be a continuous scalar field, defined on a manifold $\mathscr{M}$ that is usually a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. Given a scalar value $i \in \mathbb{R}$, called the *isovalue*, we can study the structure of a scalar field through its *level sets*, $f^{-1}(i) = \{p \in \mathscr{M} \mid f(p) = i\}$. Level sets encode the subset of $\mathscr{M}$ that is the preimage of the value i. They are also called isosurfaces, and they are the generalization of contour lines on a topographic map. Although individual level sets are often descriptive, the relationships between level sets when i changes describe important relationships.

The points of $\mathscr{M}$ where the gradient, ∇f, is equal to zero are the *critical points* of f [4]. Critical points are particularly important as a building block for studying level sets, as they correspond to values where level sets undergo topological changes. In particular, critical points may be classified by their *index* $\mathscr{I}$, which equals 0 for minima and d for maxima points. Critical points with indices between 1 and $d-1$ are called saddles.

If we imagine sweeping through all possible isovalues i, extremal critical points (minima and maxima) are positions where level set components are created and destroyed, and saddles indicate where level set components merge and split. During such a sweep, we can imbue critical points with a notion of scale. To compute this notion, we apply Elder's rule [24], which pairs critical points such that each critical point appears in only one pair (c_i, c_j) with $f(c_i) < f(c_j)$ and $\mathscr{I}(c_i) = \mathscr{I}(c_j) - 1$. The height of the pair $p = f(c_j) - f(c_i)$, called the *persistence*, encodes the life span of the level set created at $f(c_i)$ and destroyed at $f(c_j)$ [16, 25]. Persistence introduces a measure that is frequently used to distinguish topological "noise"—features that live only at small scales—from more significant, highly persistent features. More formally, the field of persistent homology studies the evolution of the homology groups characterized by critical points.

A topological abstraction known as the *persistence diagram* offers a view of the distribution of critical points of f. The persistence diagram $\mathscr{D}(f)$ embeds each pair (c_i, c_j) in the plane such that its horizontal coordinate equals $f(c_i)$, and the vertical coordinates of c_i and c_j are $f(c_i)$ and $f(c_j)$. In $\mathscr{D}(f)$, there is a vertical bar for the pair, and the height of these vertical bars is the persistence. We can also encode the distribution of critical points as a simple curve known as the *persistence curve*, which shows how many pairs of critical points in f possess a given persistence threshold or lower. Persistence diagrams and persistence curves offer high-level building blocks to see the range of scales at which features exist as well as to guide tools such as topological

simplification (Figure 3 shows an example of both).

Building further on critical points and persistence, the *Reeb graph* is a topological abstraction that segments $\mathscr{M}$ into regions where the connectivity of $f^{-1}(i)$ does not change, clustering points if they belong to the same connected component of the level set. Let $f^{-1}(f(p))_p$ be the connected component of $f^{-1}(f(p))$ containing p. The Reeb graph $\mathscr{R}(f)$ is a one-dimensional simplicial complex defined as the quotient space $\mathscr{R}(f) = \mathscr{M}/\sim$ by the equivalence relation $p_1 \sim p_2$, which holds if $p_2 \in f^{-1}(f(p_1))_{p_1}$. The contour tree (the loop-free variant of $\mathscr{R}(f)$) is often preferred in the context of visualization as it is efficient to compute [10, 64] when we can guarantee that $\mathscr{M}$ is simply connected.

Reeb graphs and contour trees have been well studied by the visualization community, offering a number of approaches for their computation in two- and three-dimensional settings [5, 17, 22, 23, 30, 54, 56, 57, 67]. In our work, we construct segmentations of the input domain based on the contour tree, and rely on the fact that these segmentations correspond to regions of interest in the data (specifically, we consider vortex identification). We rely on persistence to help us identify which portions of this segmentation are significant by helping a user navigate the space of interesting topological features.

Topological Analysis for Higher Order Data Thus far, topological analysis has been utilized for higher order data in only limited settings because the recovery of topological structures for low-order representations has presented significant computational challenges. Most techniques assume piecewise linear interpolants or utilize Forman's discrete Morse theory [28]; in both cases the combinatorial complexity of topological structures stays manageable. Notable exceptions consider quadratic [20], bilinear [52], and piecewise trilinear interpolants [1, 55]. Carr and Snoeyink propose an abstract framework for interpolants of arbitrary order [9]. In general, though, implementing a complete topological analysis is challenging for higher order interpolants because finding critical points involves determining roots of equations of degree greater than five which is analytically intractable. A notable exception is the work of Nucha et al. that considers contour tree computations for 2D piecewise polynomial functions [53]. This work advances the state-of-the-art, but it still offers only one facet of the analysis (contour trees). Most importantly, it does not consider cases where higher order discontinuous functions are used, such as HO-FEM derived fields and the discontinuous Galerkin method that we consider here. Therefore, we see our work as offering a stopgap to answer the question of how best to make use of the widely available low-order tools as approximations for complex, high-order data that suffer from the problems mentioned above. In such a setting, we cannot expect a full, verifiably correct topological extraction [26], so instead we focus on characterizing the behavior of these approximations and their effects on downstream analysis.

2.2 Data Transformation Methodologies

We refer to the mapping from discontinuous high-order polynomial data to continuous piecewise linear data as our *data transformation methodology*. Due to their computational simplicity, two simple methods are commonly used as data transformations methods. The first method is to resample the original (unstructured) data onto a regular grid. This resampling can induce minor persistent features. The second method is to subdivide the original simulation mesh hierarchically and then evaluate it. In both cases, this sampling choice can be shown mathematically to act as a filter that "removes" discontinuities from the data. Although increasing the sampling rate improves the accuracy of the filter, the effect of discontinuities in the original data also becomes more pronounced in the resultant field as we continue to add more sampling points. Therefore, we expect that it will be challenging to pick an ideal sampling rate to balance these issues.

Recently, SIAC filters have gained popularity for post-processing HO-FEM due to their ability to increase smoothness at element boundaries (i.e., remove discontinuities) while maintaining the order of accuracy, and in many cases they have been shown to improve the accuracy of the solution. Extending the work done by Bramble and Schatz [6], Cockburn et al. [14, 15] introduced SIAC filters for increasing the accuracy of discontinuous Galerkin methods with uniform spacing. Later Mirzaee et al. [45–48] introduced techniques to apply the SIAC filter at arbitrary points, thus extending its application to unstructured meshes. In [51, 58, 71], variations of the SIAC filter, called one-sided filters, are introduced to deal with boundaries and mathematical discontinuities in the solution (e.g., shocks in supersonic compressible flows). To improve accuracy on hexagonal meshes, Mirzargar et al. [49] proposed hexagonal SIAC filters. Li et al. [43] discussed effective ways to calculate the derivative quantities using SIAC and one-sided SIAC filters.

SIAC filters are also frequently used to create visualization data. Steffen et al. used SIAC filters for improving the streamline integration through the discontinuous field [63]. Walfisch et al. used 1D SIAC filters on 2D discontinuous data to create continuous streamlines [72]. Later, Docampo-Sánchez et al. proved that the Line-SIAC filter (L-SIAC), when applied to 2D and 3D simulation data, has all the properties of the SIAC filter [21], but is more computationally efficient than the traditional tensor-product-based SIAC filter in multiple dimensions. Jallepalli et al. compared the commonly used data transformation methodologies to the L-SIAC filter and showed that the L-SIAC filter is a better tool for creating continuous visualization data [37]. This work, in part, motivates us to study its effect on downstream analysis with topological tools. Recently, Jallepalli and Kirby introduced algorithms that significantly improve the computational speed of L-SIAC filter when used to create data for visualization [39].

2.2.1 L-SIAC

A Line-SIAC filter (L-SIAC) is defined as a 1D SIAC filter rotated at an angle (θ), scaled with characteristic length (H), and convolved 2D or 3D field data. Thus, the L-SIAC filter can be defined as

$$u^*(\mathbf{x}) = \int_{-\infty}^{\infty} K_H(t) * u_h(\mathbf{x} - \Gamma(t))dt$$

$$\Gamma(t) = (t\cos(\theta), t\sin(\theta)), \theta \text{ is constant}$$

where u^* is the postprocessed solution, u_h is the dG solution of degree k, H is the characteristic length defined as $h(cos(\theta)+sin(\theta))$, h is the uniform element size, and the SIAC kernel is defined along $\Gamma(t)$ by the parameter t as

$$K_H^{2k+1,k+1}(t) = \sum_{\gamma=-k}^{k} c_\gamma \Psi_H^{k+1}(t-\gamma) = \frac{1}{H}\sum_{\gamma=-k}^{k} c_\gamma \Psi^{k+1}\left(\frac{t}{H}-\gamma\right),$$

$$K_H^{2k+1,k+1}(t) = \frac{1}{H}K^{2k+1,k+1}\left(\frac{t}{H}\right).$$

For nonuniform meshes used in practical applications, dynamically adapting the characteristic length [38] based on the neighborhood of the filter produced a more accurate solution. The B-splines used in the postprocess are well studied and can be computed using the recurrence relation.

$$\Psi^1 = X_{[-1/2,1/2]},$$

$$\Psi^{k+1} = \frac{1}{k}\left(\left(t+\frac{k+1}{2}\right)\Psi^k(t+\frac{1}{2}) + \left(\frac{k+1}{2}-t\right)\Psi^k(t-\frac{1}{2})\right), k \geq 1.$$

The coefficients of the kernel, c_λ, can be found by using the property that the kernel must not destroy the accuracy of the approximation. More specifically, the coefficients reproduce polynomials of degree $2k$ by convolution. When using a symmetric B-spline kernel, we can solve the coefficients and store them for reuse. All algorithms described in this paper assume that a symmetric filter can be applied at the location where the L-SIAC filter is enforced.

The result of applying the L-SIAC filter to either a continuous or discontinuous Galerkin field $u_h(\mathbf{x})$ is an updated field $u^*(\mathbf{x})$ that is of higher degree within each element *and* that has higher levels of continuity between elements (continuity as high as $k-1$ when using a k^{th} degree filter). We can now sample this more accurate and more continuous representation of the data for input to topological analysis tools.

3 METHODS

For our empirical study, we use an analysis pipeline that transforms higher order input data to linear data, which is then analyzed through the lens of level set topology. Our pipeline has multiple parameters that we discuss in this section. We experiment with three different transformation filters in this analysis pipeline, each of which produces linear representations of scalar fields. We refer to these methods as "sampled" (for data sampled on a grid), "subdivided" (for data where we subdivide the input mesh), and "L-SIAC" (for data transformed using the L-SIAC filter). For each method, we manually select parameters that help to provide the best possible transformation.

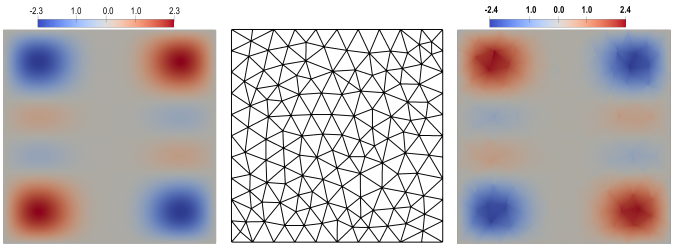
Next, we employ the Topology ToolKit (TTK) [66] to extract topological features of our filtered data using ParaView [2]. TTK requires piecewise linear representations of the input data, encoded on a simplicial mesh, and it can accept inputs as both unstructured input meshes and implicit triangulations of data sampled on regular grids. As mesh resolution is intimately related to analysis quality, we experiment with multiple possible resolutions.

Finally, our analysis pipeline involves computing an overview of features with TTK, in particular extracting critical points and computing their persistence, as well as visualizing persistence diagrams and persistence curves. We use this information to guide a segmentation of the data domain based on using the contour forests method for fast extraction of contour trees [30]. We consider multiple scales of features, based on a manual analysis of the persistence diagram to set persistence thresholds. To restrict our analysis to features at a particular scale, we perform topological simplification on the transformed data (i.e., continuous piecewise linear data), relying on the method of Tierny and Pascucci [68] to simplify the underlying scalar field and reduce noise while preserving topological features that are above a user-specified persistence threshold.

As an example to describe our analysis pipeline, we consider a simple, 2D function

$$f(x,y) = (sin(2\pi x) + \frac{1}{2}sin(4\pi x))(sin(2\pi y) + sin(4\pi y))$$

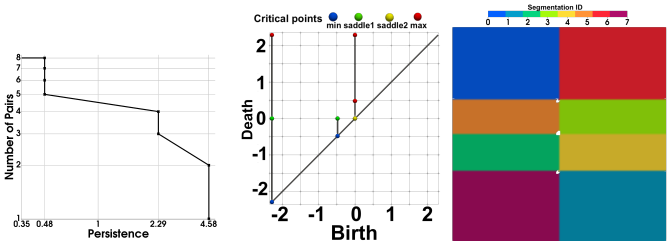
which is asymmetric in the x and y directions. It also contains topological features that persist at two different sets of scales. The first set of features with maximum persistence (two maxima and two minima) is located near the corners of the domain, and the second set of features with lower persistence is located between the first set of features along the x-axis as shown in Figure 2a. To create a ground truth comparison, we sample $f(x,y)$ at a sufficiently high resolution to capture its topological features using TTK, as shown in its persistence diagram, curve, and segmentation using the contour tree in Figure 3. This decomposes the domain into mainly 8 regions, each associated with a leaf of the contour tree for the 8 extrema. The small flat region near the center where the level sets merge at saddles accounts for the 3 small white holes (since we exclude segments associated with interior arcs).



(a) Original function (b) Simulation mesh (c) Projected data

Fig. 2: The function $f(x,y)$, simulation mesh, and the projected data.

To create a higher order representation of $f(x,y)$, we project it on an unstructured triangular element mesh (Figure 2b) over the domain $[0,1] \times [0,1]$ with degree 2 polynomial expansions on each element (hereafter referred to as the simulation mesh). With this simple example, we can mimic the elementwise discontinuities we get when computing derived fields from high-order FEM data. In the following subsections, the projected data (shown in Figure 2c) is transformed using our three different filtering methodologies to create low-order data suitable for input to TTK.



(a) Persistence curve (b) Persistence diagram (c) Segmentation

Fig. 3: The persistence curve, the persistence diagram, and the segmentation of the original function $f(x,y)$ (Figure 2a).

3.1 Transformation by Sampling to a Grid (Method: Sampled)

Our first transformation methodology is to sample the simulation data on an equispaced grid and then implicitly triangulate this grid using TTK. In this technique, sampling the projected data acts as a crude filter with only one free parameter: the sampling rate itself. This transformation thus applies no special considerations for the discontinuous piecewise polynomial data; it simply overlays a continuous piecewise linear mesh and interpolates the higher order data using the nearest element.

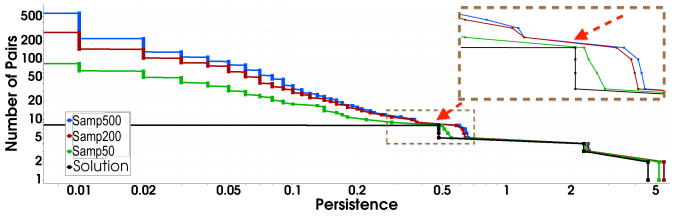
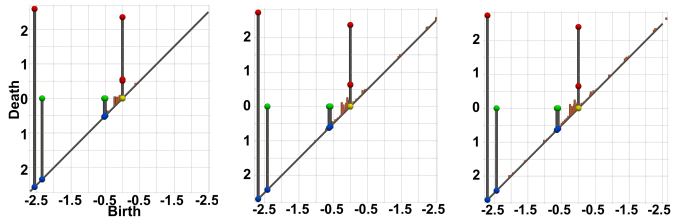


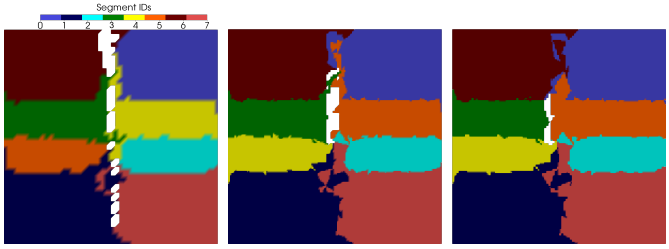
Fig. 4: Persistence curves of the sampled data (at three resolutions) as compared to the original function solution.

Figure 4 shows the persistence curves for the projected data sampled using different resolutions (50×50, 200×200, and 500×500 referred to as Samp50, Samp200, and Samp500, respectively). The average time taken to compute vorticity at each location on a machine with a 2.4 GHz (Intel CPU E7-4870) processor is 25 microseconds. We were surprised to discover that an increase in the sampling rate increased the number of low persistence pairs, which is counterintuitive to the idea that by simply using a higher resolution mesh we can approximate data better. This increase of low-persistence pairs is a direct byproduct of having additional resolution (which allows more critical points to exist), and this added resolution near the discontinuities creates low-persistence perturbation (resulting in the staircase artifacts on the left side of Figure 4). Fortunately, using the persistence curve of the actual solution as a guide, we can classify all the persistence pairs to the right of the red arrow in Figure 4 as significant features in the solution.

Typically, one searches for plateau regions in persistence curves to find stable ranges for topological simplification. For example, in the actual solution (Figure 3a), there is a stable region of 8 critical pairs to the left of 0.48, and another stable region between 0.48 and 2.29. In the area isolated by the dotted box in Figure 4, we see that the sampled data also have similar plateaus, but the plateaus persist through different ranges of persistence. For the highest resolution, this gap is slightly wider than it is for the less sampling rates, suggesting it may be easier to identify the stable region. Notably, however, between 8 and 4 pairs, all three sampled data do not drop instantaneously, but rather take four small steps. This change in stable regions suggests that, in the sample data, we have slightly perturbed the values of the minima and maxima that define these features. These observations are also highlighted in the persistence diagrams for the sampled data in Figures 5a, 5b, and



(a) Persistence Diagram: Samp50 (b) Persistence Diagram: Samp200 (c) Persistence Diagram: Samp500



(d) Segmentation: Samp50 (e) Segmentation: Samp200 (f) Segmentation: Samp500

Fig. 5: Visualizations of the persistence diagrams ((a), (b), and (c)) and segments ((d), (e), and (f)) of the sampled data at different resolutions

5c. The persistence pairs below 0.4 threshold (visualized using orange bars) increase with an increase in sampling resolution.

To segment the sampled data, we first simplify the data using the persistence threshold 0.4 units (slightly left of the red arrow in Figure 4) and then segment the data using a contour tree. Visualizations for the segmented data are shown in Figures 5d, 5e, and 5f. We observe that the boundaries of the segmentation are affected by the element boundaries of the simulation mesh, in particular in the flat region in the center vertical strip. This effect is more pronounced in highly sampled data (Samp500).

Hence, while sampling the projected data for topological analysis, it is not always the case that the highest resolution is preferred. For our experiments where ground truth is not known, we need to take care to pick a sampling resolution sufficiently high to capture the underlying features, but not too high lest we might start picking up the artifacts of discontinuities in the simulation data as features.

3.2 Transformation by Subdividing the Input Mesh (Method: Subdivided)

In practical applications, simulation meshes are usually designed to capture the variation in the simulation output, and often utilize non-uniform resolutions and unstructured alignments to better approximate interesting behaviors. To take advantage of the properties of the simulation mesh, in this technique we subdivide each element into an equal number of smaller elements to capture the underlying high-order polynomial data. On the shared boundaries between elements (e.g., the edge that bounds multiple triangles, or a vertex that is adjacent to multiple simplices), if there is a discontinuity in the higher order data, we must choose which value we use for our piecewise linear approximation. In such cases, we choose to average between different values at all adjacent elements to create continuous piecewise linear data. This technique of subdividing the elements into smaller elements is a filter that aims to both resolve the discontinuities by averaging and sample sufficiently to represent the piecewise polynomial data as continuous, piecewise linear data.

To capture the projected data on the simulation mesh, which consists of triangular elements of polynomial degree two, we subdivide each element into 9, 81, and 400 triangular elements and refer to them as Subdiv-9, Subdiv-81, and Subdiv-400, respectively. The average time taken to compute vorticity at each location on a machine with a 2.4 GHz (Intel CPU E7-4870) processor is 0.804 microseconds. Typically, each element is subdivided based on the polynomial degree (k) and the dimension (d) of the element. Each element subdivides into $(k+1)^d$

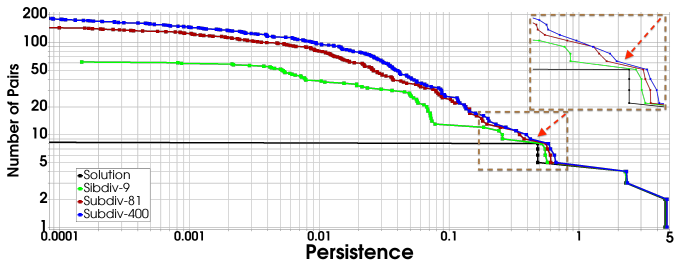
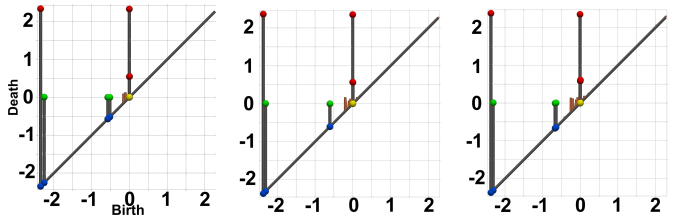
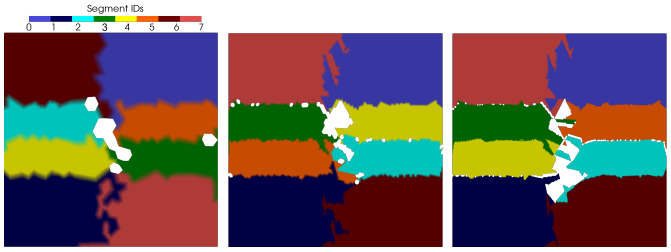


Fig. 6: Persistence curves of the subdivided data (at three resolutions) compared to the original function solution.

elements. Therefore, we consider Subdiv-9 (as $(2+1)^2 = 9$) to be the minimum amount required for subdivision. We also investigate Subdiv-81 and Subdiv-400 to see how further refinement affects the result. In all cases, we evaluate the values at new vertices internal to the elements and average the boundaries when necessary.



(a) Persistence Diagram: Subdiv-9 (b) Persistence Diagram: Subdiv-81 (c) Persistence Diagram: Subdiv-400



(d) Segmentation: Subdiv-9 (e) Segmentation: Subdiv-81 (f) Segmentation: Subdiv-400

Fig. 7: Visualizations of the persistence diagrams ((a), (b), and (c)) and segments ((d), (e), and (f)) of the subdivided data at different resolutions

Figure 6 shows the persistence curves for the projected data sampled on Subdiv-9, Subdiv-81, and Subdiv-400. Observe that an increase in the subdivisions again increases the initial number of persistence pairs, caused by the discontinuities being mapped to averages. In this case, however, the behavior appears to more smoothly vary with persistence threshold and does not have the same "staircase" artifact. Using the persistence curve of the actual solution as a guide, we examine the region associated with the first major topological change at the red arrow in Figure 6. We observe the gap between the persistence pairs due to the features and the discontinuities gap decreases, with an increase in subdivisions. Specifically, this plateau has a width of 0.26 units for Subdiv-9, 0.18 units for Subdiv-81, and 0.06 units for Subdiv-400. Similar observations are also highlighted in the persistence diagrams for the sampled data in Figures 7a, 7b, and 7c. The persistence pairs below 0.4 threshold (visualized using orange bars) increase with an increase in sampling resolution.

To segment the sampled data, we first simplify the data using the persistence threshold 0.4 units (red arrow in Figure 6) and then segment using a contour tree. Visualizations for the segmented data are shown in Figures 7d, 7e, and 7f. We observe that the boundaries of the segmentation are affected by the element boundaries of the simulation mesh, and this effect is more pronounced for the mesh with the most subdivisions (Subdiv-400).

For subdivided data as compared with sampled data, it turns out we

have a similar challenge: we need to take care in specifying the output resolution as compared to the size of features we are interested in. For our remaining experiments, which do not have ground truth, we use the resolution of $(k+1)^d$, which is the minimum resolution required to capture the underlying features. We also remark that while averaging appears to provide a mathematical fix for discontinuities, it comes with the cost of potentially distorting the shapes of the segmentation.

3.3 Transformation by Applying the L-SIAC Filter (Method: L-SIAC)

For our third transformation method, we use the L-SIAC filter to post-process the solution and define new values across the domain. We can then sample the resulting output to a regular grid, using resolutions h, which are equivalent to the grid resolutions we used in the sampled data. For applying the L-SIAC filter at a point, we first choose the parameters: the characteristic length of the L-SIAC filter is adapted based on the size on the element [38], the angle is chosen to be 0 degrees, the order of B-splines equal to $k+1=3$ (k is degree of the element), and the number of B-splines is $2k+1=5$. To apply the L-SIAC filter at a point in the mesh, we shift the filter to that point and then project the L-SIAC filter onto the mesh to find the overlapping elements. We then use Gaussian quadrature to integrate the L-SIAC kernel with the underlying mesh. In this technique, the L-SIAC filter and sampling frequency work together to control the conversion process.

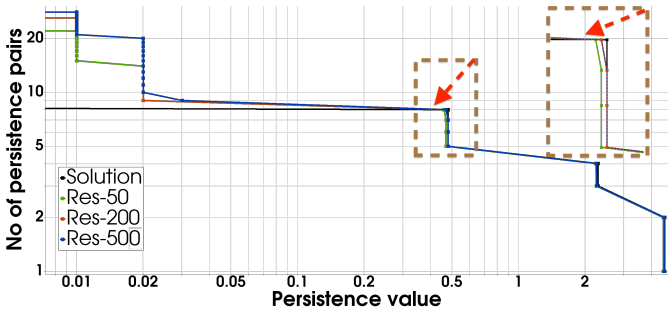
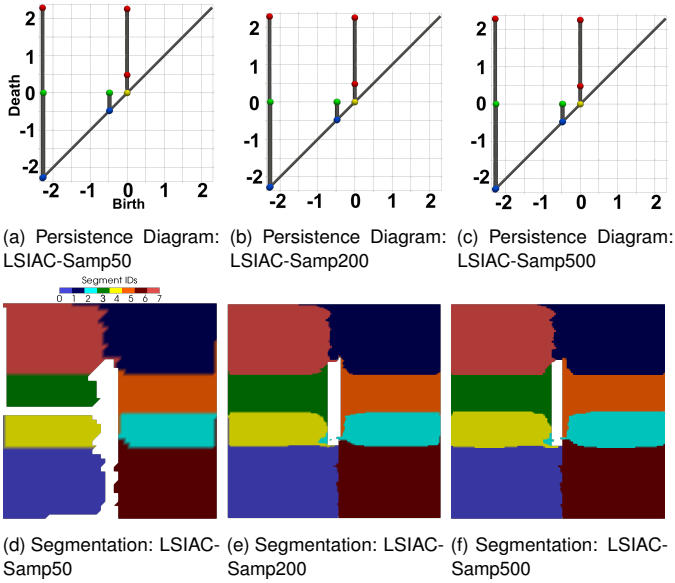


Fig. 8: Persistence curves of the L-SIAC data (at three resolutions) as compared to the original function solution.

Figure 8 shows the persistence curves for the projected data sampled using different resolutions (50×50, 200×200, and 500×500 referred to as LSIAC-Samp50, LSIAC-Samp200, and LSIAC-Samp500, respectively). The average time taken to compute vorticity at each location on a machine with a 2.4 GHz (Intel CPU E7-4870) processor is 334 microseconds. Although L-SIAC mitigates most of the effect of discontinuities in the data, we still have some aberrant features that show up at low persistence values due to its error compared to the actual solution. However, we observe that these features disappear significantly early, creating a wider stable region that begins at far lower persistence thresholds. Additionally, the sampling rate has a negligible influence on when this region starts and ends, and the behavior at larger persistence values almost matches the ground truth data. These observations are also confirmed in the persistence diagrams in Figures 9a, 9b, and 9c.

To segment the sampled data, we first simplify the data using the persistence threshold 0.4 units (red arrow in Figure 8) and then segment it using a contour tree. Note, however, we could use a much lower persistence threshold than before because L-SIAC appears to mitigate topological noise that was present in the sampled and subdivided data. Visualizations for the segmentation from the contour tree are shown in Figures 9d, 9e, and 9f. Although we observe the same number of persistent features at all resolutions, they differ at coarse resolutions, at which the shapes of these regions can vary. In particular, we see more boundary artifacts for coarser data at the nearly flat region in the center of the domain. Higher resolutions help to recover these boundaries more smoothly. This recovery suggests that for more complex data, increasing the resolution of L-SIAC filter is helpful in terms of both the number and shape of features, albeit only as shown for this simple dataset.



(a) Persistence Diagram: LSIAC-Samp50 (b) Persistence Diagram: LSIAC-Samp200 (c) Persistence Diagram: LSIAC-Samp500

(d) Segmentation: LSIAC-Samp50 (e) Segmentation: LSIAC-Samp200 (f) Segmentation: LSIAC-Samp500

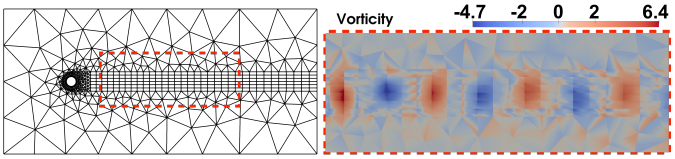
Fig. 9: Visualizations of the persistence diagrams ((a), (b), and (c)) and segments ((d), (e), and (f)) of the L-SIAC data at different resolutions

4 Results

In this section, we present both two-dimensional and three-dimensional results. For our first example, we analyze the results of simulating flow past a two-dimensional circular cylinder. In our second example, we analyze a three-dimensional flow scenario that produces a collection of co-rotating vortices.

4.1 Flow Over a 2D Cylinder

The Nektar++ [8] solver suite, and in particular the incompressible Navier-Stokes solver, was used to generate the fluid flow results. Flow past a circular cylinder at the viscosity examined is a transient problem, but here we analyze only a single snapshot (in time) topologically. The mesh used for this simulation is shown in Figure 10a, which contains polynomial degree two ($P(2)$) elements: 500 triangles and 330 quadrilaterals found primarily in the wake region behind the circle. A continuous Galerkin (FEM) methodology was used that has degree 2 inside the element and has C^0 continuity at the element interfaces. The Reynolds number was set at $Re=500$, and the flow solver was run until shedding behind the cylinder generated consistently shaped vortices.



(a) Simulation mesh. (b) Vorticity.

Fig. 10: (a) Simulation mesh of the flow over a 2D cylinder. (b) Vorticity calculated using element derivatives in the highlighted region of the simulation mesh.

The simulation data from the highlighted region of the simulation mesh (Figure 10) was chosen to be analyzed topologically. We analyze the scalar field of vorticity. To calculate the vorticity, we need to apply a derivative on the C^0 vector field components (u,v); thus creating discontinuity at the element boundaries as shown in Figure 10b. From this dataset, we use the methods described in Section 3 to produce three different low-order datasets for topological analysis.

To produce our "sampled" vorticity (Section 3.1), we sampled the simulation output on a regular grid of resolution 145×80 and calculated partial derivatives (u_y, v_x) using a finite difference scheme. We then used these derivatives to compute the vorticity defined as $\omega_z = v_x - u_y$.

To produce our "subdivided" vorticity (Section 3.2), we subdivided the simulation mesh such that each triangle was divided into 9 triangles, and each quadrilateral was divided into 18 triangles. The simulation data (u,v) was sampled and stored at the vertices of the subdivided mesh. To calculate the partial derivatives, we first calculated the derivatives at the center of each triangle using the values at its vertices, and then the derivatives were interpolated back to the vertices using a weighted area of all the triangles connected to each vertex. To produce the L-SIAC vorticity (Section 3.3), we used the derivative L-SIAC filter $(D(K(3,6))$, where D stands for derivative) as presented in [37]. This filter calculates the partial derivatives (u_y, v_x) at any point, and can be evaluated on a regular grid of resolution 145×80. The average time taken to compute vorticity at each location on a machine with a 2.4 GHz (Intel CPU E7-4870) processor is 86, 0.496, and 560 microseconds for the sampled, subdivided, and L-SIAC vorticities, respectively.

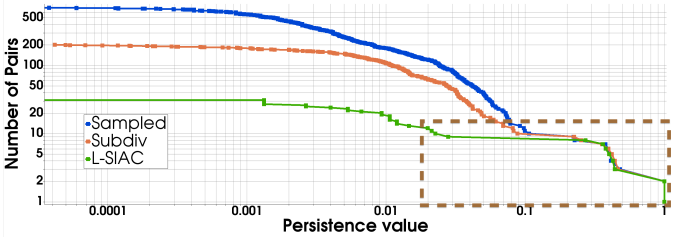


Fig. 11: Persistence curves for the sampled, subdivided, and L-SIAC vorticity fields of the flow over a 2D cylinder.

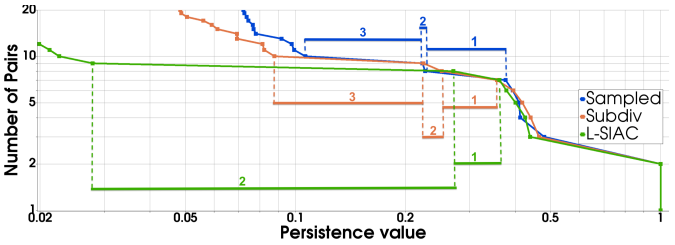
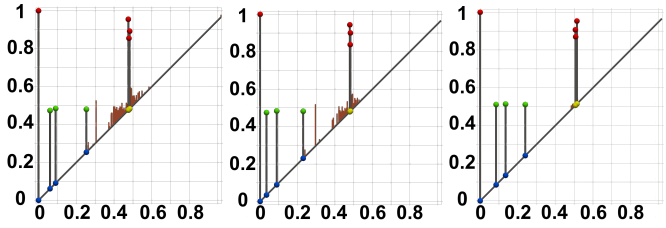


Fig. 12: Zoom in on the region associated with Figure 11. For each of datasets, we highlight two different stable regimes of critical features (numbered 1-3).

Due to different approximation errors while calculating the vorticity, each scalar field has a different range. Specifically, the range of values is smallest for the L-SIAC ($[-3.85, 3.69]$), and the subdivided vorticity ($[-4.36, 4.67]$) has a smaller range than that of the sampled vorticity ($[-4.80, 5.22]$). To enable comparison between the three datasets, we normalized their ranges to $[0,1]$. The persistence curves for the vorticity tests calculated above are shown in Figure 11. We observe that the sampled vorticity and the subdivided vorticity have a large number of persistence pairs compared to the L-SIAC vorticity. The function of the persistence curve is to act as a guide to help choose the persistence threshold required to identify the features and simplify (remove) the noise. Key persistence thresholds are identified by detecting the plateau regions in the persistence curve. From the persistence curves in Figure 12, we identified three regions of interest for the sampled and subdivided (indicated by 1, 2, and 3), and two regions in case of L-SIAC (indicated by 1 and 2). The regions indicated by 1, 2 and 3 have 7, 8, and 9 persistence pairs. As it turns out, there are only 8 significant features in the dataset (determined by inspecting the data). The plateau corresponding to the 8 features indicated by region 2 is significantly shorter in the case of sampled and subdivided vorticity compared to that in the L-SIAC filter, and also shorter than its counterparts (regions indicated by 1 and 3).

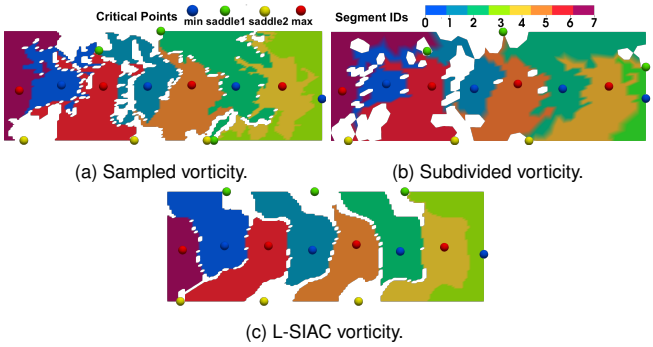
We pick a persistence value in the regions indicated by 2 in all the datasets (i.e., 0.225 for the sampled, 0.23 for the subdivided, and 0.2 in the case of the L-SIAC vorticity) to create the persistence diagrams shown in Figure 13. The persistence pairs below the threshold are



(a) Sampled vorticity. (b) Subdivided vorticity. (c) L-SIAC vorticity.

Fig. 13: The persistence diagrams of the vorticity for the fluid flow past a circular cylinder described in Section 4.1. Thick gray lines are used to show the persistence pairs above the threshold 2.0, and the lines in orange are used to indicate the persistence pairs below the threshold.

shown by orange lines. Observe that in the case of subdivided and sample vorticity, there exists a persistence pair (the tallest orange bar) that is very close to the threshold but does not exist in the case of the L-SIAC vorticity. This feature corresponds to a boundary artifact that creates a spurious but high-persistence, feature that would be hard to separate without additional knowledge.



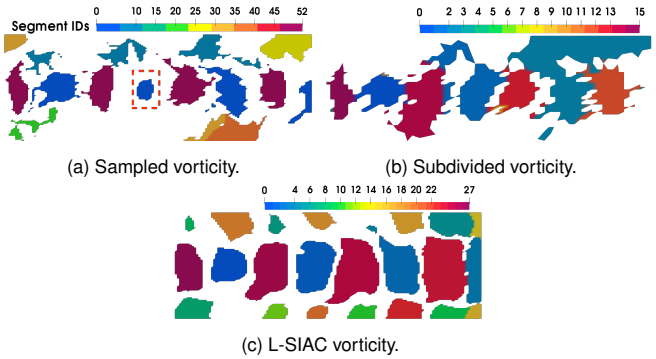
(a) Sampled vorticity. (b) Subdivided vorticity.

(c) L-SIAC vorticity.

Fig. 14: Segmentation of the vorticity over a flow past a cylinder described in Section 4.1. The segmentations are calculated using the contour tree and a persistence threshold in region 2 of Figure 12. The critical points denoted by saddle1 and saddle2 are the saddles corresponding the merges and splits, respectively.

The segmentation of the datasets along with the location and type of critical points are shown in Figure 14, created using the contour tree and the persistence threshold in region 2 (the same ones used to create the persistence diagrams in Figure 13). The segmentation in all three cases captures and classifies the significant part of the vortices. Observe the critical points of the max and mins lineup at similar locations for the three datasets. The critical points, however, for saddle1 and saddle2 appear more uniformly distributed in the case of the L-SIAC vorticity. The nonuniformly distributed critical points for saddle1 and saddle2 affects the boundaries of the segmented regions for the sampled and subdivided vorticity, which are far more irregular and exhibit resemblance to the orientations in the simulation mesh. In the case of L-SIAC vorticity, the edges of the segmentations are smoother and align only with the mesh boundaries.

In an attempt to segment the vortices more stringently without collapsing features, we applied the contour tree using lower persistence thresholds to oversegment the domain. We used persistence threshold of 0.04 to the sampled and the subdivided vorticity, thus resulting in the segmentation shown in Figures 15a and 15b, respectively. In the case of the L-SIAC vorticity, we chose the persistence threshold of 0.001, which contains all the persistence pairs (refer to Figure 11) and used it for segmentation (Figure 15c). To select the segmentations containing the vortices, we filtered them by thresholding based on their size and checking if the segment is associated with a leaf (a minima or maxima) in the contour tree. In the case of sampled vorticity (Figure 15a), a portion of the vortex indicated by the red dotted box was filtered out. In this case also (referring to Figure 15), we observed that the boundaries of segmentations for sampled and subdivided vorticity are irregular



(a) Sampled vorticity. (b) Subdivided vorticity.

(c) L-SIAC vorticity.

Fig. 15: Segmentation of the vorticity for the flow past a cylinder described in Section 4.1. The segmentation for the sampled and the subdivided vorticity are computed using the contour tree along with a persistence threshold of 0.04. In the case of L-SIAC vorticity, the value of persistence threshold used is 0.001.

compared to the boundaries of the segments in the L-SIAC vorticity.

4.2 Counter-Rotating Vortex

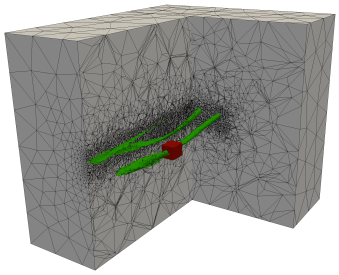


Fig. 16: Input simulation of the counter-rotating vortices. We focus on analyzing data in the red cube. The counter-rotating vortices in green are iso-surfaces of vorticity (magnitude) calculated using element-wise derivatives.

We also consider a 3D example for our empirical study, again using a Nektar++ simulation example. An incompressible Navier-Stokes solver with cG discretization was used to generate a flow scenario containing two primary vortices and their secondary counter-rotating vortices shown in Figure 16. The simulation (input) parameters were set to the following: for the advection term, the Velocity Correction Scheme [40] with SVV de-aliasing was used, and the time integration splitting scheme was set to IMEX order two. Further details on the simulations are given in [60]. The simulation mesh was adaptively refined at the location of the vortices and contains $223,837$ polynomial degree five ($P(5)$) tetrahedra.

The simulation data from the highlighted region in Figure 16 (indicated by the red cube) was chosen to be analyzed topologically. We analyzed the magnitude of vorticity as the scalar quantity to extract vortices. We used the elementwise derivatives on the vector field quantities (u,v,w) to calculate the magnitude of the vorticity vector (hereafter referred to as the vorticity). To produce the "sampled" vorticity, elementwise derivatives are sampled on a grid of resolution of $100\times100\times100$ and used to calculate the magnitude of vorticity. To produce "subdivided" vorticity, each tetrahedron of the input mesh, in the highlighted region was subdivided into smaller tetrahedrons($216=6^3$). The vorticity was sampled at the vertices of the subdivided mesh and at the vertices having a discontinuity, the values were averaged. To calculate the L-SIAC vorticity, we use L-SIAC methodology proposed in Jallepalli et al. [37]. All the required derivatives are calculated using the L-SIAC filter. The parameters used for the L-SIAC filter are B-splines of order 7 (specifically, we use the $D^1K(11,7)$ filter, where D^1 represents the total derivative), the θ in direction of the derivative and the characteristic length is adapted based on the element size [38]. The average time taken to compute vorticity at each location on a machine with a 2.5 GHz (Intel CPU E7-8890) processor is 0.34, $9.6e^{-4}$, and 334 milliseconds for the sampled, subdivided, and L-SIAC vorticities, respectively.

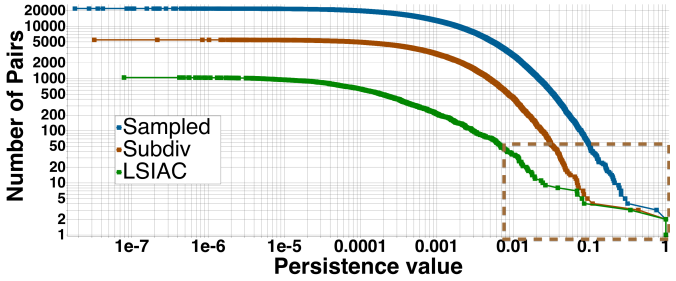


Fig. 17: Persistence curves for the sampled, subdivided, and L-SIAC vorticity fields of the counter-rotating vortex.

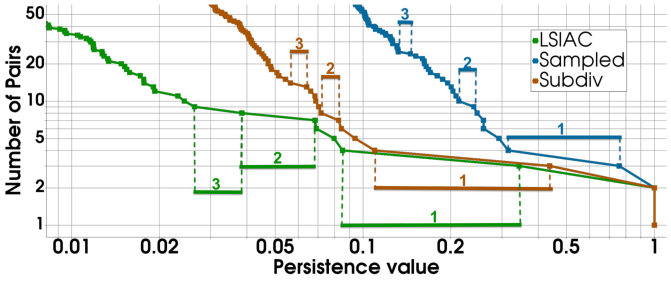
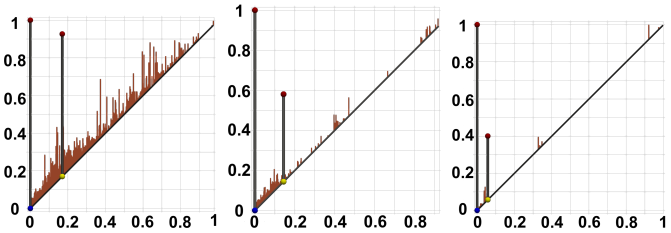


Fig. 18: Zoom in on the region associated with Figure 17. For each of datasets, we highlight three different stable regimes of critical features (numbered 1-3).

The datasets had vorticity ranging from $[0,84]$, $[0,88]$, and $[0,176]$ for sampled, subdivided and L-SIAC vorticity fields, respectively. To enable comparisons of persistence curves and diagrams, we normalized the vorticity to the same range of $[0,1]$. The persistence curves are shown in Figure 17. We observe that the first stable regions indicated by 1 in Figure 18, are have ranges of persistence at $[0.32,0.76]$, $[0.11,0.44]$, and $[0.08,0.34]$ for the sampled, subdivided, and L-SIAC vorticity. The sampled vorticity has the most extended stable region, followed by subdivided and then the L-SIAC filter. Note that the ranges appear differently due to the log-scaled x-axis in Figure 18.

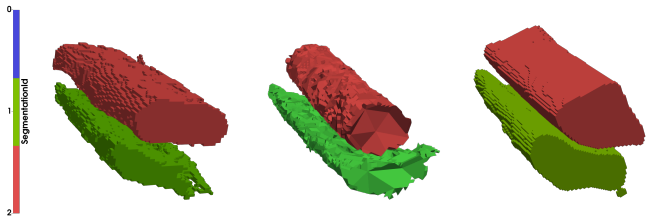


(a) Sampled vorticity at persistence threshold 0.5. (b) Subdivided vorticity at persistence threshold 0.2. (c) L-SIAC vorticity at persistence threshold 0.2.

Fig. 19: Persistence diagrams for the vorticity of the counter-rotating vortex dataset. Orange bars highlight what was removed by topological filtering.

Consequently, we chose different ranges of persistence values for topological simplification. We first considered a coarse simplification into the stable region 1. The persistence diagrams simplified using thresholds of 0.5 (sampled) and 0.2 (subdivided and L-SIAC) in the respective stable regions are shown in Figures 19. To highlight what was removed, the persistence pairs below the threshold are visualized as orange bars. Unlike our other examples, we observed that sampled vorticity has a substantial number of persistence pairs far from the

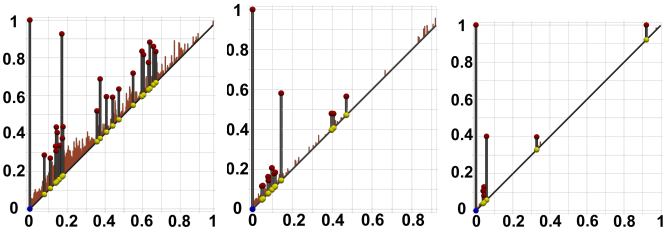
diagonal – suggesting that there was significant topological "noise" in the data that makes it more difficult to separate even the most prominent vortices of interest. By comparison, the subdivided and L-SIAC datasets appear to have more separation between the persistent features.



(a) Sampled vorticity. (b) Subdivided vorticity. (c) L-SIAC vorticity.

Fig. 20: Segmentation corresponding to the topologically simplified vorticity fields as the threshold used in Figure 19. These figures highlight the two main rotating vortices, as best described by each dataset.

Using the persistence threshold in the stable region 1 (0.5 for sampled, 0.2 subdivided and L-SIAC) to simplify the dataset, we used the contour trees to segment the domain and visualized the segmented vortex cores in Figure 20. We observed that all three methods had segmented significant portions of the primary and secondary vortices. The boundaries of the sampled and subdivided vorticity were slightly irregular compared to the L-SIAC filter. However, at this coarse scale, they were not significantly worse. The subdivided mesh is expected to have irregularities due to the underlying nonuniform mesh. Thus, all the methods have identified and segmented the first and second significant features of interest in the dataset. This result demonstrates that for this dataset, topological simplification can tolerate a wide range of transformation methodologies to capture the coarsest scale of features.



(a) Sampled vorticity at persistence threshold 0.136 (b) Subdivided vorticity at persistence threshold 0.06. (c) L-SIAC vorticity at persistence threshold 0.03.

Fig. 21: Persistence diagrams for the vorticity of the counter-rotating vortex dataset after simplifying to capture fine-scale features.

To identify a second set of features from the data, we looked for smaller persistence thresholds in the datasets. In Figure 18, among the stable regions indicated by 2 and 3, the stable region indicated by 3 in all the datasets yielded the best possible segmentation of the data. The persistence diagrams using the thresholds in the stable region 3 (0.136 for sampled, 0.06 for subdivided, 0.03 for L-SIAC) are shown in Figure 21.

The lower threshold we selected to segment the data resulted in numerous undesired segmentations of the data that we removed during analysis. Specifically, we manually identified segments using their level in the contour tree, size of the segment, and the segment IDs, for segments that resembled vortex-like structures. The resulting segmentations from the datasets are shown in Figure 1. We observe that the sampled vorticity (Figure 1a) produced 4 of 45 segments resembling vortex-like structures, but the vortex indicated by blue was divided into smaller undesirable regions. In the case of subdivided vorticity (Figure 1b), we identified 3 of 23 segments that resembled vortices, but the segmented region indicated in green has many gaps, meaning significant parts of the vortex is segmented into smaller segments that have been filtered out. This was particularly surprising given the structure in the persistence diagram for subdivided vorticity, which appears to have less topological noise. In the case of L-SIAC vorticity (Figure 1c), we identified 5 of 13 segments resembling vortices that segment the vortices consistently, except for the segment represented by blue. A portion of this vortex surrounds its neighboring vortices.

To summarize, the sampled, the subdivided, and the L-SIAC methodologies successfully identified the most significant features in the dataset. To identify the fine-scale features, the L-SIAC methodology performed best among the three, but we remark that it comes at a significant computational cost relative to merely sampling [37]. The sampled vorticity appeared to capture the shape of the vorticity features better than the subdivided vorticity. Furthermore, it also identified an extra vortex than the subdivided vorticity. However, the sampled data has the disadvantage that filtering to capture the best set of segments required removing a larger number of features.

5 Discussion

The precise characterization of the effects of data transformations on topological analysis can manifest in subtle ways, as small changes in function values can affect the order in which features merge. In this empirical study, we take a first step toward understanding the relationships among element discontinuities, sampling artifacts, and topological features. We analyzed three methodologies that can be used to enable topological analysis of the HO-FEM data. For identifying the most significant features in HO-FEM data, we used two simple methodologies: sampling on a grid and subdividing the mesh. While both have the advantage of simplicity, and both are successful at capturing the coarsest features, they also tend to distort the size and shape of the feature boundaries. In particular, subdivision appears to separate features better, but sampling might do a better job of capturing the shape of the feature. Interestingly, we discovered and demonstrated the counterintuitive nature of these methodologies, since in certain ways they perform adversely with the increase in sampling resolution.

We have also shown that an extended range of features can be identified in the dataset using the L-SIAC filter. Using the L-SIAC filter, we can improve the resolution of the features with the increase in sampling resolution. Although the L-SIAC filter may appear better suited to this analysis task, it is also computationally expensive as its cost of evaluation is roughly proportional to the sampling frequency. In future work, it would be interesting to use topological analysis on sampled or subdivided data as a guide to adaptively choose locations to apply the L-SIAC filter and use the updated results to improve the topological analysis iteratively. In particular, at regions that are far from discontinuities, we expect all methodologies to perform better.

The current focus of this work has been on scalar fields and their topological analysis through persistence diagrams and contour trees. A limitation of this study is its scope. In the future, we would like to broaden this study to include gradient field topology (through Morse-Smale complexes) as well as the topology of vector and tensor data. Nevertheless, the narrow scope we have employed has helped to isolate specific issues with both interpolants and feature extraction techniques. Long term, we think this study can help motivate a better set of design constraints for directly extracting topological features from higher order data without the need for data transformation. That said, even current tools appear capable of extracting certain coarse features if data transformation is used carefully.

Acknowledgments

The authors also wish to thank Professor Spencer Sherwin (Imperial College London, UK), Mr. Alexandre Sidot, and the Nektar++ Group for the counter-rotating vortex data and helpful discussions. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number(s) DE-SC-0019039. The authors acknowledge support from ARO W911NF-15-1-0222 (Program Manager Dr. Mike Coyle).

References

[1] A. Acharya and V. Natarajan. A parallel and memory efficient algorithm for constructing the contour tree. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 271–278. IEEE, 2015.

[2] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large-data visualization. *The Visualization Handbook*, pp. 717–731, 2005.

[3] F. Al Sulti and K. Ohkitani. Vortex merger and topological changes in two-dimensional turbulence. *Physical Review E*, 86(1):016309, 2012.

[4] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 1970.

[5] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 2008.

[6] J. H. Bramble and A. H. Schatz. Higher order local accuracy by averaging in the finite element method. *Mathematics of Computation*, 31(137):94–111, 1977.

[7] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

[8] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. M. Kirby, and S. J. Sherwin. Nektar++ : An open-source spectral/hp element framework. *Computer Physics Communications*, 192:205–219, Jul 2015. doi: 10.1016/j.cpc.2015.02.008

[9] H. Carr and J. Snoeyink. Representing interpolant topology for contour tree computation. In *Topology-Based Methods in Visualization II*, pp. 59–73. Springer, 2009.

[10] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[11] H. Carr, J. Snoeyink, and M. van de Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, 43(1):42 – 58, 2010.

[12] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In *Symp. on Comp. Geom.*, 2003.

[13] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient morse decompositions of vector fields. *IEEE Trans. Visualization and Computer Graphics*, 14(4), 2008.

[14] B. Cockburn, M. Luskin, C.-W. Shu, and E. Süli. Post-processing of Galerkin methods for hyperbolic problems. In *Discontinuous Galerkin Methods*, pp. 291–300. Springer, 2000.

[15] B. Cockburn, M. Luskin, C.-W. Shu, and E. Súli. Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Mathematics of Computation*, 72(242):577–606, 2003.

[16] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Symp. on Comp. Geom.*, 2005.

[17] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *Symp. on Comp. Geom.*, pp. 344–350, 2003.

[18] M. Day, J. Bell, P.-T. Bremer, V. Pascucci, V. Beckner, and M. Lijewski. Turbulence effects on cellular burning structures in lean premixed hydrogen flames. *Combustion and Flame*, 156:1035–1045, 2009.

[19] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. *Comp. Grap. For.*, 2015.

[20] S. E. Dillard, V. Natarajan, G. H. Weber, V. Pascucci, and B. Hamann. Topology-guided tessellation of quadratic elements. *International Journal of Computational Geometry & Applications*, 19(02):195–211, 2009.

[21] J. Docampo-Sánchez, J. K. Ryan, M. Mirzargar, and R. M. Kirby. Multi-dimensional filtering: Reducing the dimension through rotation. *SIAM Journal on Scientific Computing*, 39(5):A2179–A2200, 2017.

[22] H. Doraiswamy and V. Natarajan. Output-sensitive construction of reeb graphs. *IEEE Trans. on Vis. and Comp. Graph.*, 2012.

[23] H. Doraiswamy and V. Natarajan. Computing reeb graphs as a union of contour trees. *IEEE Trans. on Vis. and Comp. Graph.*, 2013.

[24] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009.

[25] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Disc. Compu. Geom.*, 2002.

[26] T. Etiene, L. Nonato, C. Scheidegger, J. Tienry, T. Peters, V. Pascucci, R. Kirby, and C. Silva. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):952–965, June 2012. doi: 10.1109/TVCG.2011.109

[27] N. Fehn, W. A. Wall, and M. Kronbichler. Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows. *Journal of Computational Physics*, 372:667–693, 2018.

[28] R. Forman. A user's guide to discrete Morse theory. *Adv. in Math.*, 1998.

[29] D. Guenther, R. Alvarez-Boto, J. Contreras-Garcia, J.-P. Piquemal, and J. Tierny. Characterizing molecular interactions in chemical systems. *IEEE Trans. on Vis. and Comp. Graph.*, 2014.

[30] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Contour forests: Fast multi-threaded augmented contour trees. In *IEEE LDAV*, 2016.

[31] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study in combustion. *Comp. Graph. For.*, 2014.

[32] A. Gyulassy, P. T. Bremer, B. Hamann, and V. Pascucci. A practical approach to morse-smale complex computation: Scalability and generality. *IEEE Trans. on Vis. and Comp. Graph.*, 2008.

[33] A. Gyulassy, A. Knoll, K. Lau, B. Wang, P. Bremer, M. Papka, L. A. Curtiss, and V. Pascucci. Interstitial and interlayer ion diffusion geometry extraction in graphitic nanosphere battery materials. *IEEE Trans. on Vis. and Comp. Graph.*, 2015.

[34] A. Gyulassy, V. Natarajan, M. Duchaineau, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically Clean Distance Fields. *IEEE Trans. on Vis. and Comp. Graph.*, 2007.

[35] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Comp. Grap. For.*, 2016.

[36] Z. W. H.T. Huynh and P. Vincent. High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids. *Computers & Fluids*, 98:209–220, 2014.

[37] A. Jallepalli, J. Docampo-Sánchez, J. K. Ryan, R. Haimes, and R. M. Kirby. On the treatment of field quantities and elemental continuity in FEM solutions. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):903–912, 2017.

[38] A. Jallepalli, R. Haimes, and R. M. Kirby. Adaptive characteristic length for L-SIAC filtering of fem data. *Journal of Scientific Computing*, pp. 1–22, 2018.

[39] A. Jallepalli and R. M. Kirby. Efficient algorithms for the line-siac filter. *Journal of Scientific Computing*, 2019.

[40] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp element methods for Computational Fluid Dynamics (Second Edition)*. Oxford University Press, 2005.

[41] J. Kasten, J. Reininghaus, I. Hotz, and H. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Trans. on Vis. and Comp. Graph.*, 2011.

[42] D. E. Laney, P. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. on Vis. and Comp. Graph.*, 2006.

[43] X. Li, J. K. Ryan, R. M. Kirby, and C. Vuik. Smoothness-Increasing Accuracy-Conserving (SIAC) filters for derivative approximations of discontinuous Galerkin (DG) solutions over nonuniform meshes and near boundaries. *Journal of Computational and Applied Mathematics*, 294:275–296, 2016.

[44] J.-E. W. Lombard, D. Moxey, S. J. Sherwin, J. F. A. Hoessler, S. Dhandapani, and M. J. Taylor. Implicit large-eddy simulation of a wingtip vortex. *AIAA Journal*, 54(2):506–518, 2016.

[45] H. Mirzaee, L. Ji, J. K. Ryan, and R. M. Kirby. Smoothness-increasing accuracy-conserving (SIAC) postprocessing for discontinuous Galerkin solutions over structured triangular meshes. *SIAM Journal on Numerical Analysis*, 49(5):1899–1920, 2011.

[46] H. Mirzaee, J. King, J. K. Ryan, and R. M. Kirby. Smoothness-increasing accuracy-conserving filters for discontinuous Galerkin solutions over unstructured triangular meshes. *SIAM Journal on Scientific Computing*, 35(1):A212–A230, 2013.

[47] H. Mirzaee, J. K. Ryan, and R. M. Kirby. Efficient implementation of smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions. *Journal of Scientific Computing*, 52(1):85–112, 2012.

[48] H. Mirzaee, J. K. Ryan, and R. M. Kirby. Smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions: Application to structured tetrahedral meshes. *Journal of Scientific Computing*, 58(3):690–704, 2014.

[49] M. Mirzargar, A. Jallepalli, J. K. Ryan, and R. M. Kirby. Hexagonal smoothness-increasing accuracy-conserving filtering. *Journal of Scientific Computing*, 73(2-3):1072–1093, 2017.

[50] D. Moxey, R. Amici, and R. M. Kirby. Efficient matrix-free high-order finite element evaluation for simplicial elements. *SIAM Journal on Scientific Computing*, Under Review, 2019.

[51] D.-M. Nguyen and J. Peters. Nonuniform discontinuous Galerkin filters via shift and scale. *SIAM Journal on Numerical Analysis*, 54(3):1401–1422, 2016.

[52] G. Norgard and P.-T. Bremer. Robust computation of morse–smale complexes of bilinear functions. *Computer Aided Geometric Design*, 30(6):577–587, 2013.

[53] G. Nucha, G.-P. Bonneau, S. Hahmann, and V. Natarajan. Computing contour trees for 2d piecewise polynomial functions. In *Computer Graphics Forum*, vol. 36, pp. 23–33. Wiley Online Library, 2017.

[54] S. Parsa. A deterministic o(m log m) time algorithm for the reeb graph. In *Symp. on Comp. Geom.*, 2012.

[55] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(1):249–268, 2004.

[56] V. Pascucci, G. Scorzelli, P. T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. on Graph.*, 2007.

[57] G. Patane, M. Spagnuolo, and B. Falcidieno. Reeb graph computation based on a minimal contouring. In *Proc. of IEEE Shape Modeling International*, 2008.

[58] J. K. Ryan, X. Li, R. M. Kirby, and K. Vuik. One-sided position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering over uniform and non-uniform meshes. *Journal of Scientific Computing*, 64(3):773–817, 2015.

[59] N. Shivashankar, P. Pranav, V. Natarajan, R. van de Weygaert, E. P. Bos, and S. Rieder. Felix: A topology based framework for visual exploration of cosmic filaments. *IEEE Trans. on Vis. and Comp. Graph.*, 2016. `http://vgl.serc.iisc.ernet.in/felix/index.html`.

[60] A. Sidot. *Spectral/hp Element Methods Applied To Vortex Structures In A Low Incidence Delta Wing Wake Compared To Experiments*. M.S. Thesis in Aeronautics, Imperial College London, September 2017.

[61] B. S. Sohn and C. L. Bajaj. Time varying contour topology. *IEEE Trans. on Vis. and Comp. Graph.*, 2006.

[62] T. Sousbie. The persistent cosmic web and its filamentary structure: Theory and implementations. *Royal Astronomical Society*, 2011. `http://www2.iap.fr/users/sousbie/web/html/indexd41d.html`.

[63] M. Steffen, S. Curtis, R. M. Kirby, and J. K. Ryan. Investigation of smoothness-increasing accuracy-conserving filters for improving streamline integration through discontinuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):680–692, 2008.

[64] S. Tarasov and M. Vyali. Construction of contour trees in 3d in o(n log n) steps. In *Symp. on Comp. Geom.*, 1998.

[65] D. M. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Trans. on Vis. and Comp. Graph.*, 2014.

[66] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The topology toolkit. *IEEE Trans. Vis. Comput. Graph.*, 24(1):832–842, 2018.

[67] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Trans. on Vis. and Comp. Graph.*, 2009.

[68] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE Trans. on Vis. and Comp. Graph.*, 2012.

[69] X. Tricoche, C. Garth, G. Kindlmann, E. Deines, G. Scheuermann, M. Ruetten, and C. Hansen. Visualization of intricate flow structures for vortex breakdown analysis. In *IEEE Visualization 2004*, pp. 187–194. IEEE, 2004.

[70] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pasucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Symp. on Comp. Geom.*, 1997.

[71] P. van Slingerland, J. K. Ryan, and C. Vuik. Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving discontinuous Galerkin solutions. *SIAM Journal on Scientific Computing*, 33(2):802–825, 2011.

[72] D. Walfisch, J. K. Ryan, R. M. Kirby, and R. Haimes. One-sided smoothness-increasing accuracy-conserving filtering for enhanced streamline integration through discontinuous fields. *Journal of Scientific Computing*, 38(2):164–184, 2009.

[73] G. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Trans. on Vis. and Comp. Graph.*, 2007.