

Extracting Complex Topology from Multivariate Functional Approximation: Contours, Jacobi Sets, and Ridge-Valley Graphs

Guanqun Ma*
University of Utah

David Lenz†
Argonne National Laboratory

Hanqi Guo‡
Ohio State University

Tom Peterka§
Argonne National Laboratory

Bei Wang¶
University of Utah

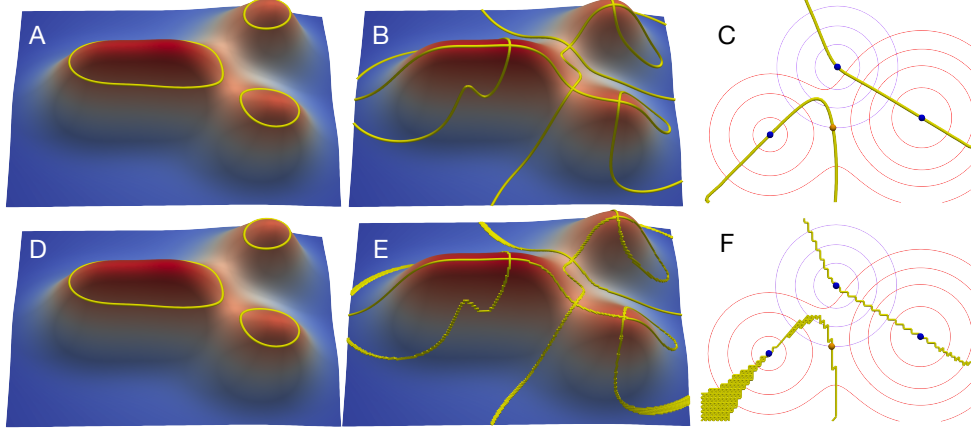


Figure 1: Extracting complex topological descriptors (highlighted in yellow) from MFA models. First, given an MFA model as input, we extract contours at a fixed isovalue in (A) and a ridge-valley graph in (B) directly from the MFA model, without discretizing the entire domain. Next, for a pair of MFA models, we extract the Jacobi set in (C), where the contours from each model are shown in purple and red, respectively. Finally, we compare these topological descriptors with the contour (D), ridge-valley graph (E), and Jacobi set (F) derived from discrete representations of the MFA models. Local maxima and saddles are indicated by blue and orange dots in (C) and (F), respectively.

ABSTRACT

Implicit continuous models, such as functional models and implicit neural networks, are an increasingly popular method for replacing discrete data representations with continuous, high-order, and differentiable surrogates. These models offer new perspectives on the storage, transfer, and analysis of scientific data. In this paper, we introduce the first framework to directly extract complex topological features—contours, Jacobi sets, and ridge-valley graphs—from a type of continuous implicit model known as multivariate functional approximation (MFA). MFA replaces discrete data with continuous piecewise smooth functions. Given an MFA model as the input, our approach enables direct extraction of complex topological features from the model, without reverting to a discrete representation of the model. Our work is easily generalizable to any continuous implicit model that supports the queries of function values and high-order derivatives. Our work establishes the building blocks for performing topological data analysis and visualization on implicit continuous models.

Index Terms: Implicit continuous model, topological data analysis, contour, Jacobi set, ridge-valley graph.

1 INTRODUCTION

Implicit continuous models—functional models (e.g., spline representations and radial basis functions) and implicit neural networks (e.g., sinusoidal representation networks [49])—have gained growing interest in scientific data analysis and visualization. These implicit models replace discrete data representations with continuous,

high-order, and differentiable data representations, thus improving the flexibility and efficiency of scientific workflows [30, 33, 49] and enabling complex analysis of scientific data [30, 36, 50].

A notable example of an implicit continuous model is the multivariate functional approximation (MFA) [39, 40]. An MFA model represents discrete data by approximating it with a set of piecewise smooth polynomial functions. It allows for the evaluation of both function values and high-order derivatives at any point within the domain. By utilizing a mesh-free, uniform representation, MFA enables the remapping between different data representations and supports spatiotemporal analysis in a continuous setting. Supported by the SciDAC RAPIDS Institute [2] of the U.S. Department of Energy (DOE), MFA has found applications in high-energy physics [1] and climate science [3]. It has proven effective as an intermediate data representation for high-quality volume rendering [51].

Despite recent advancements, we are only beginning to explore the potential of implicit continuous models in supporting topological data analysis and visualization. Topological descriptors have been instrumental in enabling a number of scientific visualization tasks, such as feature detection, tracking, and clustering; see [17, 57] for surveys. However, existing topological methods work with data represented on discrete grids. Naively applying these methods to implicit continuous models requires discretization of the entire domain. However, this discretization effectively reduces the high-order model to a set of low-order features, which can introduce aliasing artifacts.

We therefore tackle a key question: Given a continuous model as the input, such as an MFA, how can we extract topological descriptors *directly* from the model without resorting to discretization of the entire domain? Recently, Ma et al. [29] extracted critical points directly from an MFA model without discretization. Building on [29], we present a method for directly extracting more complex features from MFA models: contours, Jacobi sets and ridge-valley graphs.

*e-mail: guanqun.ma@utah.edu

†e-mail: dlennz@anl.gov

‡e-mail: guo.2154@osu.edu

§e-mail: tpeterka@mcs.anl.gov

¶e-mail: beiwang@sci.utah.edu

Topological descriptors such as contours, ridge-valley graphs, and Jacobi sets are important in studying scientific data. Given a scalar function, a contour (or level set) represents a part of its domain that remains a constant value. Contours thus provide insights into the structural information of data at user-specified thresholds [27]. A ridge-valley graph delineates ridges and valleys of a scalar function that are invariant under translations, rotations, uniform magnification, and monotonic transformations [37]. The Jacobi set, on the other hand, arises from multiparameter data analysis: given a pair of scalar functions, it identifies points where their gradients are aligned. In particular, the Jacobi set reveals the relationship between a pair of functions by analyzing the critical points of one function restricted to the contour of another [14].

The framework presented in this paper takes full advantage of the continuous and differentiable representations afforded by MFA models. Our contributions are summarized below.

- We present a framework for extracting contours directly from continuous models—specifically, MFA models—without discretizing the entire domain. We utilize particle tracing along directions perpendicular to the gradient to accurately locate contours, by leveraging the ability of these models to query function values and derivatives at any point in the domain.
- We extract Jacobi sets and ridge-valley graphs from MFA models by converting the problem to contour extraction from derived functions. This approach simplifies and unifies the extraction process across multiple topological descriptors.

2 RELATED WORK

Functional models and implicit neural networks. Implicit continuous models represent scalar functions in a smooth and differentiable manner [7, 32]. Unlike discrete representations that may suffer from limited resolution, continuous models can be evaluated at any point in the domain. They support queries of function values and high-order derivatives, which are essential for data analysis and visualization [16, 29].

Implicit neural networks—a family of continuous models—have gained significant attention in scientific machine learning [38, 49, 58]. These models enable the representation of complex shapes, scenes, and signals without explicit discretization. One prominent example of an implicit neural network is the sinusoidal representation network (SIREN) [49]. SIREN utilizes periodic activation functions for implicit neural representations (INR) and has been used in volumetric data compression [28].

Multivariate functional approximation. MFA has emerged as a powerful method for representing scientific datasets using continuous B-spline functions (see e.g., [24, 39, 40]). An MFA model could be considered as a form of scattered data approximation (SDA) that builds continuous functions to approximate spatial datasets [56]. Several SDA methods have been proposed with functional approximations based on splines [11], wavelets [20], and radial basis functions [30].

An MFA model distinguishes itself from other SDA methods by leveraging the properties of B-splines to achieve high-order continuity and efficient computation. It reduces storage due to its compact representation and supports the computation of function values and derivatives of any order at any point in the domain [24]. Lenz et al. [24] used MFA to enable customizable approximations of complex datasets. Sun et al. [51, 52] developed scalable volume visualization techniques based on MFA representations, demonstrating high-quality rendering with reduced computational overhead.

Ma et al. [29] introduced critical point extraction from an MFA model. As critical points are the simplest forms of topological features, their work could be considered as a first step toward extracting complex topology from MFA.

Contour extraction is a fundamental task in scientific visualization, allowing the exploration of complex functions by identifying

curves or surfaces along which the function matches a user-specified constant [27]. Classic methods use the marching cubes algorithm [27] and its variants [25, 35] to extract contours from discrete data on grids. For 3D (resp. 2D) data, the marching cubes (resp. marching squares) algorithm examines how each contour passes through a cube/square. Ju et al. [21] presented a dual contouring algorithm to extract contours from a signed grid, utilizing gradient information to improve the extraction quality. While classic methods are based on discrete data, continuous implicit models give accurate evaluation of function values and derivatives. As shown in this paper, utilizing implicit models could bypass the data discretization step and improve accuracy.

Our approach is inspired by methods used in surface–surface intersection for CAD models, with the distinction that one of the surfaces is derived from the gradient of the other. It also relates to the literature on non-isolated root finding. For example, Dokken [12] employed recursive subdivision techniques to compute intersections between a B-spline surface and a plane. While his method is tailored to B-splines, our approach is adaptable to more general classes of functions. Polynomial homotopy continuation [5] provides an alternative framework for solving systems of equations. Theisel et al. [53] tracked critical points in a 2D time-dependent vector field by integrating streamlines of a derived field. In a manner similar to our method, they used a Runge–Kutta integration scheme to ensure that the resulting critical lines are independent of an underlying grid.

Jacobi sets capture the relationships among multiple scalar functions by identifying points where their gradients are aligned [14]. Edelsbrunner and Harer [14] formalized the concept of Jacobi sets for multiple Morse functions, exploring the topological properties and applications. Jacobi sets and Reeb spaces are commonly used topological descriptors for multiparameter data analysis. A Reeb space captures the structure of a multiparameter mapping by compressing the connected components of its level sets. Jacobi sets and Reeb spaces are connected through a mapping between their singularities [8]. Such a mapping was studied by Chattopadhyay et al. [9]. Klötzl et al. utilized a local bilinear method to approximate a Jacobi set with good accuracy [22], and proposed a topological connection method for Jacobi set computation [23], improving visual clarity while preserving topological structure.

Tierny and Carr [54] built on Jacobi set to compute the Reeb space of a bivariate function defined on a tetrahedral mesh. Sharma and Natarajan [47] used Jacobi sets to identify fiber surfaces. Meduri et al. [31] proposed Jacobi set simplification for tracking topological features in time-varying scalar functions.

A closely related concept is the Pareto set, which studies scalar functions based on consensus or disagreement among their critical points, ascending and descending manifolds, and connectivity [19]. Huettnerberger et al. [18] explored the relationship between Pareto sets and Jacobi sets, establishing subset and equivalence relations between them.

Compared with discrete representations, continuous implicit models enable precise identification of Jacobi sets without artifacts from data discretizations. Leveraging the continuous and differentiable nature of MFA models could improve the efficiency and accuracy of Jacobi set extraction, benefiting multiparameter data analysis and visualization.

Ridge-valley graphs. Damon [10] proposed ridge-valley-connector-curves that describe the global structure of height ridges. A point belongs to a height ridge if the scalar function has a local maximum in the direction orthogonal to its gradient. Norgard and Bremer introduced the ridge-valley graph [37] based on the notion of Jacobi ridges, which behave similarly to the curves proposed in [10]. Jacobi ridges are points where the gradient magnitude has a local minimum along the level set [37]. However, unlike height ridges [13], Jacobi ridges are invariant under monotonic

transformations, although their structures are similar and their formulations coincide for quadratic functions [37]. A ridge-valley graph is the Jacobi set of a function and its squared gradient magnitude (see Sec. 3).

Later studies introduced different methods for extracting ridges. For instance, Anisotropic Gaussian Kernel (AGK) is employed to enhance sensitivity and robustness [26]. Reisenhofer and King [46] refined local features by integrating the contrast-invariant phase congruency measure with α -molecules.

3 TECHNICAL BACKGROUND

We provide the background of MFA, contours, Jacobi sets, and ridge-valley graphs to support the understanding of our method.

3.1 Multivariate Functional Approximation

MFA models are tensor-product B-spline functions that approximate a discrete dataset. We briefly review B-splines below; see [11, 42] for a detailed presentation of B-spline theory.

A degree p B-spline curve is a piecewise-polynomial function. It has $p - 1$ continuous derivatives and is composed of degree p polynomial pieces. The junction points between these pieces are referred to as *knots*, and the intervals between knots are *knot spans* or simply *spans*. The shape of the curve is determined by a set of *control points* distributed across the domain. Although the curve loosely follows a polyline formed by these control points, it does not necessarily pass through them; see Fig. 2.

Mathematically, a B-spline F is defined as a combination of *basis functions* N_j , weighted by the control point positions P_j :

$$F(u) = \sum_{j=0}^{n-1} N_j(u) P_j. \quad (1)$$

In 1D, given a set of points located at $\{u_0, \dots, u_{m-1}\} \subset [0, 1]$, where each u_i is associated with a value f_i , the best-fit B-spline curve has optimal control points to minimize the pointwise error:

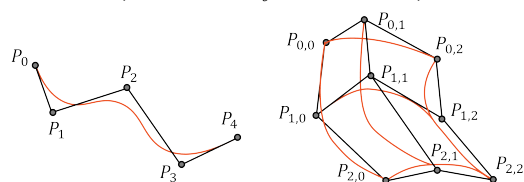
$$\min_P \left(\frac{1}{m} \sum_{i=0}^{m-1} \left| f_i - \sum_{j=0}^n N_j(u_i) P_j \right|^2 \right)^{1/2}. \quad (2)$$


Figure 2: A 1D B-spline curve (left) and a 2D B-spline surface (right). P_i and $P_{i,j}$ are control points; control meshes are in black; approximated curves or surfaces are in red.

In this paper, we focus on MFA in the 2D case. A 2D B-spline surface of degree p is defined using two sets of knots, $\{t_{j_1}^{(1)}\}_{j_1=0}^{n_1+p}$ and $\{t_{j_2}^{(2)}\}_{j_2=0}^{n_2+p}$, corresponding to the u and v dimensions, respectively. $\{P_{j_1, j_2}\}_{j_1=0}^{n_1-1, j_2=0}^{n_2-1}$ is the set of control points. The 2D basis functions are constructed as the product of 1D basis functions in each dimension: $N_{j_1, j_2}(u, v) = N_{j_1, p}^{(1)}(u) N_{j_2, p}^{(2)}(v)$, where $N_{j_1, p}^{(1)}(u)$ and $N_{j_2, p}^{(2)}(v)$ are 1D basis functions for the u and v dimensions. The resulting 2D B-spline surface is expressed as:

$$F(u, v) = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} N_{j_1, p}^{(1)}(u) N_{j_2, p}^{(2)}(v) P_{j_1, j_2}. \quad (3)$$

Finally, a 2D span is simply the tensor product of 1D spans. For example, a 2D span corresponds to a region:

$$[t_{j_1}^{(1)}, t_{j_1+1}^{(1)}] \times [t_{j_2}^{(2)}, t_{j_2+1}^{(2)}].$$

3.2 Critical Point Extraction From an MFA Model

Since an MFA model is a piecewise-polynomial function, critical points can be directly extracted from an MFA model following the approach outlined by Ma et al. [29]:

1. Span filtration: Each span is constrained within the convex hull of its corresponding control points. By analyzing the control points of the first derivatives, we identify and exclude spans that are impossible to contain critical points, thereby reducing the spans we actually work with.
2. Critical point extraction: For the remaining spans, we use Newton's method to compute critical points in each span. The iterative update formula is given by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - H(\mathbf{x}_n)^{-1} \nabla f(\mathbf{x}_n), \quad (4)$$

where $H(\mathbf{x}_n)$ is the Hessian matrix and $\nabla f(\mathbf{x}_n)$ is the gradient of the MFA model f evaluated at \mathbf{x}_n . Since MFAs are differentiable functions, each of these queries is exact.

3. Deduplication: To ensure uniqueness, we remove duplicated critical points using spatial hashing.

According to [29], the overall time complexity in a 2D domain is $\mathcal{O}(i_{max} p^4 n)$, where i_{max} is the maximum number of iterations using Newton's method and n is the number of spans.

3.3 Particle Tracing

Particle tracing involves finding the trajectories of particles moving through a vector field, essential in visualizing flow patterns such as streamlines and pathlines [41]. The motion of a particle can be described by an initial value problem (IVP) in the form of an ordinary differential equation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(t, \mathbf{x}), \mathbf{x}(t_0) = \mathbf{x}_0, \quad (5)$$

where $\mathbf{x}(t)$ represents the position of a particle at time t and $\mathbf{v}(t, \mathbf{x})$ is the velocity at position \mathbf{x} and time t . Particle tracing is performed by numerically solving the IVP [43] in (5). To do so, numerical integration methods such as the Runge-Kutta method are used.

The classic Runge-Kutta method (RK4) [44] is a fourth-order method to approximate \mathbf{x}_{n+1} from \mathbf{x}_n in (5):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{s}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \quad (6)$$

$$t_{n+1} = t_n + s, \quad (7)$$

where $\mathbf{k}_1 = \mathbf{v}(t_n, \mathbf{x}_n)$, $\mathbf{k}_2 = \mathbf{v}(t_n + \frac{s}{2}, \mathbf{x}_n + \frac{s}{2} \mathbf{k}_1)$, $\mathbf{k}_3 = \mathbf{v}(t_n + \frac{s}{2}, \mathbf{x}_n + \frac{s}{2} \mathbf{k}_2)$, and $\mathbf{k}_4 = \mathbf{v}(t_n + s, \mathbf{x}_n + s \mathbf{k}_3)$, and $s > 0$ is the step size. RK4 yields \mathbf{x}_{n+1} as an approximation to \mathbf{x} at time t_{n+1} . The local truncation error per step is $\mathcal{O}(s^5)$ and the total accumulated error is $\mathcal{O}(s^4)$ [44].

3.4 Jacobi Set

Studying natural phenomena often involves analyzing the relationship between two functions defined on the same domain, such as temperature distribution within a layer of constant salinity [14]. Jacobi sets capture this relationship by analyzing the critical points of one function restricted to the contours of the other.

A function is a Morse function if all its critical points are non-degenerate and have distinct function values. Given two Morse functions $f, g : \mathbb{M} \rightarrow \mathbb{R}$ defined on a manifold \mathbb{M} , the Jacobi set $\mathbb{J} = \mathbb{J}(f, g) = \mathbb{J}(g, f)$ is defined as the closure of the set of points where the gradients of f and g are linearly dependent [14]:

$$\mathbb{J}(f, g) = \text{cl}\{\mathbf{x} \in \mathbb{M} \mid \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0 \text{ or } \nabla g(\mathbf{x}) + \lambda \nabla f(\mathbf{x}) = 0\}, \quad (8)$$

for some $\lambda \in \mathbb{R}$. This condition implies that

$$\mathbb{J}(f, g) = \text{cl}\{\mathbf{x} \in \mathbb{M} \mid \mathbf{x} \text{ is a critical point of } f + \lambda g \text{ or } g + \lambda f\}, \quad (9)$$

for some $\lambda \in \mathbb{R}$ [14]. Alternatively, let $g^{-1}(t)$ denote the level set of g at $t \in \mathbb{R}$, and let $f_t : g^{-1}(k) \rightarrow \mathbb{R}$ be the restriction of f on $g^{-1}(t)$, then $\mathbb{J}(f, g)$ can equivalently be defined [6] as

$$\mathbb{J}(f, g) = \text{cl}\{\mathbf{x} \in \mathbb{M} \mid \mathbf{x} \text{ is a critical point of } f_t\}. \quad (10)$$

The critical points of f_t are known as the *restricted critical points* of f with respect to g . Symmetrically, we can define restricted critical points of g with respect to f .

The Jacobi set $\mathbb{J}(f, g)$ is an embedded 1-manifold in \mathbb{M} (i.e., a set of curves) [14]. It includes all critical points of f and g . The function f_t is a Morse function except at a finite set of measure zero, where f_t changes in criticality at critical points of g [6]. The alignment of restricted critical points changes at critical points of f and g [6]. Examples of Jacobi sets are illustrated in Fig. 1 (C) and (F). The Jacobi set $\mathbb{J}(f, g)$ is the zero level set of the comparison measure $\kappa_x = \|\nabla f(x) \times \nabla g(x)\|$ [34].

3.5 Ridge Valley Graph

Ridges and valleys are highly sought-after features across diverse fields, from image processing [48] to combustion simulations [45]. A ridge-valley graph is a complete description of all ridges and valleys in a scalar function $f : \mathbb{M} \rightarrow \mathbb{R}$ [37]. It refers to Jacobi ridges that satisfy all five desired invariants [37]. A ridge-valley graph of f corresponds to a non-generic Jacobi set $\mathbb{J}(f, \|\nabla f\|^2)$ formed by f and its squared gradient magnitude [37]. Following [37], a ridge-valley graph of f consists of nodes and arcs with the following properties (see Fig. 1 (A)):

1. Arcs: Smoothly embedded curves in \mathbb{M} .
2. Nodes: Valence-4 nodes correspond to the critical points of f . Valence-2 nodes are the points where classification changes (see below).
3. Classification: An arc is consistently classified as a (pseudo-) ridge or a (pseudo-) valley. The classification depends on the behavior of $\|\nabla f\|^2$ along the contours of f and the behavior of f in the direction tangent to these contours. A ridge (resp. valley) point is one where $\|\nabla f\|^2$ is minimal along a contour and f is maximal (resp. minimal) tangent to the contour. A pseudo-ridge (resp. pseudo-valley) point is one where $\|\nabla f\|^2$ is maximal along a contour and f is maximal (resp. minimal) tangent to the contour.

A ridge-valley graph is invariant under translations, rotations, and uniform magnification in the spatial variables. Additionally, it is defined locally and remains invariant under monotonic transformations of f [37]. Examples of ridge-valley graphs are illustrated in Fig. 1 (B) and (E).

4 METHOD

To the best of our knowledge, we present the first framework for extracting contours, Jacobi sets, and ridge-valley graphs from continuous implicit models. We demonstrate our framework using MFA models; however, it is broadly applicable and can be adapted to any implicit model that allows querying of function values and higher-order derivatives.

4.1 Contour Extraction

Given a Morse function $f : \mathbb{M} \rightarrow \mathbb{R}$ on a 2-manifold \mathbb{M} , a *contour* (level set) γ_a of f defined at a given threshold (isovalue) $a \in \mathbb{R}$ is given by $\gamma = \gamma_a := \{\mathbf{x} \in \mathbb{M} \mid f(\mathbf{x}) = a\}$. At any point $\mathbf{x} = (x_1, x_2)^\top \in \gamma$, the gradient is orthogonal to γ , meaning the tangent direction \mathbf{m} of γ at \mathbf{x} is perpendicular to $\nabla f(\mathbf{x})$. Let $\nabla f =$

$(f_{x_1}, f_{x_2})^\top$ represent the gradient, then

$$\mathbf{m} = \frac{1}{\|\nabla f\|} (-f_{x_2}, f_{x_1})^\top, \quad (11)$$

where $\|\nabla f\| = \sqrt{f_{x_1}^2 + f_{x_2}^2}$ is the gradient magnitude.

Contour extraction can be formulated as a particle tracing problem, where a particle moves along the contour by following the tangent direction \mathbf{m} . To numerically integrate this trajectory, we employ the RK4 method by setting $\mathbf{v} = \mathbf{m}$ and ignoring time (see Sec. 3.3). Let $s > 0$ be the step size.

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{s}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \quad (12)$$

where $\mathbf{k}_1 = \mathbf{m}(\mathbf{x}_n)$, $\mathbf{k}_2 = \mathbf{m}(\mathbf{x}_n + \frac{s}{2}\mathbf{k}_1)$, $\mathbf{k}_3 = \mathbf{m}(\mathbf{x}_n + \frac{s}{2}\mathbf{k}_2)$, and $\mathbf{k}_4 = \mathbf{m}(\mathbf{x}_n + s\mathbf{k}_3)$. Starting from an initial point \mathbf{p}_0 on the contour γ , we obtain a sequence of points $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots\}$ by iteratively applying the RK4 method, thereby tracing the contour.

Our contour extraction method comprises three main steps:

1. Initialization: Identify a set of starting points on the contour.
2. Particle tracing: Utilize the RK4 method to trace the contour from each starting point.
3. Connecting: Incorporate critical points and connect the traced points to form continuous segments of the contour.

We discuss these steps in detail in Secs. 4.1.1 to 4.1.3.

4.1.1 Initialization

To find points satisfying $f(\mathbf{x}) = a$ for a given a , we convert the problem into a root-finding problem. We define $f_a(\mathbf{x}) := f(\mathbf{x}) - a$ and search for the roots of f_a by setting $f_a = 0$. We use a normalized gradient descent method to solve the root-finding problem, where the iteration formula is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \frac{\nabla f_a(\mathbf{x}_n)}{\|\nabla f_a(\mathbf{x}_n)\|}, \quad (13)$$

with an adaptive step size $\alpha = \frac{f_a(\mathbf{x}_n)}{\|\nabla f_a(\mathbf{x}_n)\|}$.

In the context of MFA, the domain is partitioned into spans, each representing a polynomial function. Within each span, we uniformly sample $(p+3)^d$ initial points, where p is the polynomial degree and $d = 2$ is the dimension. From each initial point, we apply the normalized gradient descent to find a starting point satisfying $f(\mathbf{x}) = a$ (or equivalently, $f_a(\mathbf{x}) = 0$). To avoid redundancy, we remove starting points that are closer than s to other starting points (duplication removal). The root-finding process is confined within the current span. Since each span is processed independently, this step is naturally adapted to multithreading.

4.1.2 Particle Tracing

In the particle tracing step, we continue to treat each span independently. Within a given span, we trace a piece of the contour (a trajectory) using RK4 method (see Eq. (12)) from each starting point obtained from the initialization step (Sec. 4.1.1). The procedure is as follows:

1. **Forward tracing:** Trace the trajectory forward following \mathbf{m} .
2. **Boundary handling:** If the trajectory exits within the current span, stop tracing. To make sure the distance between the last point inside the span and the first point in an adjacent span does not exceed s , we set $s \leftarrow \frac{s}{2}$ and process with RK4 one more time within the boundary.
3. **Backward tracing:** To capture the entire trajectory passing through the starting point, if the trajectory does not form a loop, we also trace backward from the starting point in the opposite direction $-\mathbf{m}$. If the trajectory returns to the starting point (indicating that it forms a loop), backward tracing is unnecessary.

When a trajectory reaches a critical point, the gradient magnitude of f becomes negligible, and an RK4 update cannot proceed effectively. In such cases, we stop tracing when $\|\nabla f(\mathbf{x}_n)\|$ is smaller than a threshold. In practice, as the trajectory approaches a critical point, the step size $\|\mathbf{x}_{n+1} - \mathbf{x}_n\|$ decreases significantly. Therefore, in all the experiments, we stop tracing when $\|\mathbf{x}_{n+1} - \mathbf{x}_n\| < 0.5s$. To prevent error accumulation during particle tracing, we utilize a correction step at every point. If $|f(\mathbf{x}_n) - a| > \epsilon$, where ϵ is the threshold to control the accuracy, we adjust the point position using the normalized gradient descent described in Eq. (13).

Since multiple starting points may trace the same trajectory, duplication removal is necessary. Even when we stop tracing near critical points, it is possible for trajectories to skip over these critical points, leading to multiple recordings of the same trajectory. We deal with various duplication scenarios as illustrated in Fig. 3. If the distance between points from two different trajectories is less than s , we consider these points as *corresponding*. The key idea behind duplication removal is identifying duplicated segments by searching for corresponding endpoints of two trajectories.

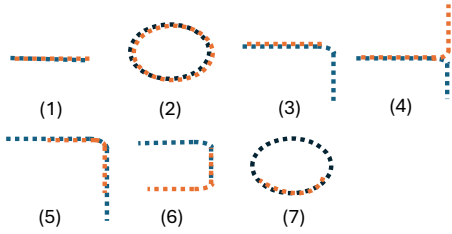


Figure 3: Illustrations of duplicated trajectories as candidates for duplication removal. Two trajectories that contain duplicated segments are shown in orange and blue respectively.

4.1.3 Connecting

After removing duplications, we connect trajectories to form continuous contours.

1. **Inserting critical points:** Following Sec. 3.2, we identify and insert critical points \mathbf{x} satisfying $\|f(\mathbf{x}) - t\| < \epsilon$. These critical points can have valence > 2 . Since particle tracing alone cannot guarantee to find these points, we add them manually to ensure completeness.
2. **Connecting to critical points:** For trajectories within a span, we connect their endpoints to nearby critical points if they are within a distance $\leq 2s$. On a contour, the valence of a critical point may be ≥ 2 , whereas that of a regular point is 2, since the gradient is nonzero at a regular point. Consequently, if a trajectory contains more than one point, it can only connect to another trajectory or a critical point.
3. **Connecting trajectories across spans:** To avoid duplication, we only compare trajectories with those in adjacent spans with higher span indices. If endpoints from different spans are within a distance of $2s$, we connect them.
4. **Connecting trajectories within a span:** We connect endpoints of trajectories within the same span if their distance is $< 2s$.
5. **Connecting within a trajectory:** Within each trajectory, we connect all points sequentially for continuity.

4.2 Jacobi Set Extraction

Following Eq. (8), a point $\mathbf{x} = (x_1, x_2)^\top$ belongs to the Jacobi set $\mathbb{J} := \mathbb{J}(f, g)$ if one of the following conditions holds: $\nabla f(\mathbf{x}) = 0$, $\nabla g(\mathbf{x}) = 0$, or the gradients $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are aligned (parallel). As noted in [15, 34], such points lie on the zero level set of $\|\nabla f(\mathbf{x}) \times \nabla g(\mathbf{x})\|$. We take inspiration from this formulation. In a 2D domain, the gradients are given by $\nabla f = (f_{x_1}, f_{x_2})^\top$ and $\nabla g = (g_{x_1}, g_{x_2})^\top$. To utilize the cross

product, we extend these gradients to 3D by adding a zero component: $\nabla f_3 = (f_{x_1}, f_{x_2}, 0)^\top$ and $\nabla g_3 = (g_{x_1}, g_{x_2}, 0)^\top$. To identify $\mathbf{x} \in \mathbb{J}$, the following cross product must be zero:

$$\nabla f_3(\mathbf{x}) \times \nabla g_3(\mathbf{x}) = (0, 0, f_{x_1}g_{x_2} - f_{x_2}g_{x_1})^\top = (0, 0, 0)^\top. \quad (14)$$

This condition implies that either $\nabla f(\mathbf{x}) = 0$, $\nabla g(\mathbf{x}) = 0$, or $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are parallel, which is consistent with the definition in Eq. (8). We then define

$$h = f_{x_1}g_{x_2} - f_{x_2}g_{x_1}. \quad (15)$$

Extracting the Jacobi set \mathbb{J} becomes equivalent to finding the contour of h with $a = 0$ (i.e., the 0-contour). We then apply the contour extraction described in Sec. 4.1 to determine the Jacobi set.

In MFA, f and g are constructed with the same span settings. Their only difference lies in their control points. As f and g share the same spans, h is also a polynomial function within each span. Consequently, our Jacobi set extraction process can be seamlessly integrated with our contour extraction method that treats each span independently. In the final step, we extract the critical points of h .

To utilize the contour extraction method, we require the gradient of h given by:

$$\nabla h = \begin{pmatrix} h_{x_1} \\ h_{x_2} \end{pmatrix} = \begin{pmatrix} f_{x_1}g_{x_1x_2} + g_{x_2}f_{x_1x_1} - f_{x_2}g_{x_1x_1} - g_{x_1}f_{x_1x_2} \\ f_{x_1}g_{x_2x_2} + g_{x_2}f_{x_1x_2} - f_{x_2}g_{x_1x_2} - g_{x_1}f_{x_2x_2} \end{pmatrix}. \quad (16)$$

In practice, we extract the critical points \mathbf{x} of h that satisfy $\|h(\mathbf{x})\| < \epsilon$ and incorporate them into the result.

4.3 Ridge-Valley Graph Extraction

Following Sec. 3.5, a ridge-valley graph is a special type of Jacobi set $\mathbb{J}(f, \|\nabla f\|^2)$. Let $g = \|\nabla f\|^2$. The gradient of g is given by $\nabla g = 2H\nabla f$, where H is the Hessian matrix of f .

Formally, H and ∇g are derived as follows:

$$H = \begin{pmatrix} f_{x_1x_1} & f_{x_1x_2} \\ f_{x_1x_2} & f_{x_2x_2} \end{pmatrix}, \quad (17)$$

$$\nabla g = 2H\nabla f = 2 \begin{pmatrix} f_{x_1x_1}f_{x_1} + f_{x_1x_2}f_{x_2} \\ f_{x_1x_2}f_{x_1} + f_{x_2x_2}f_{x_2} \end{pmatrix}. \quad (18)$$

The Hessian matrix of g , denoted by H_g , is given by

$$H_g = \begin{pmatrix} g_{x_1x_1} & g_{x_1x_2} \\ g_{x_1x_2} & g_{x_2x_2} \end{pmatrix}, \quad (19)$$

where

$$\begin{aligned} g_{x_1x_1} &= 2(f_{x_1x_1}^2 + f_{x_1x_1x_1}f_{x_1} + f_{x_1x_1x_2}f_{x_2} + f_{x_1x_2}^2), \\ g_{x_1x_2} &= 2(f_{x_1x_1x_2}f_{x_1} + f_{x_1x_1x_2}f_{x_2} + f_{x_1x_2x_2}f_{x_2} + f_{x_1x_2}f_{x_2x_2}), \\ g_{x_2x_2} &= 2(f_{x_1x_2x_2}f_{x_1} + f_{x_1x_2}^2 + f_{x_2x_2}^2 + f_{x_2x_2x_2}f_{x_2}). \end{aligned} \quad (20)$$

We can express the condition for the ridge-valley graph as:

$$\begin{aligned} \tilde{h} &= f_{x_1}g_{x_2} - f_{x_2}g_{x_1} \\ &= 2(f_{x_1}(f_{x_1x_2}f_{x_1} + f_{x_2x_2}f_{x_2}) - f_{x_2}(f_{x_1x_1}f_{x_1} + f_{x_1x_2}f_{x_2})) \\ &= 2((f_{x_1}^2 - f_{x_2}^2)f_{x_1x_2} + f_{x_1}f_{x_2}(f_{x_2x_2} - f_{x_1x_1})) = 0. \end{aligned} \quad (21)$$

Similar to the Jacobi set extraction, since \tilde{h} is derived from f and its derivatives, the span setting remains the same. We can thus apply the contour extraction method by directly substituting f and its derivatives with \tilde{h} and its derivatives to extract the ridge-valley graph. In the final step, according to [37], instead of computing the critical points of \tilde{h} where $\tilde{h} = 0$, we compute the critical points of

f and incorporate them into the result. This approach ensures that all critical points of f are included in the ridge-valley graph and it is more efficient than computing critical points of h directly. The gradient $\nabla \tilde{h}$ can be obtained by substituting Eq. (20) into Eq. (16).

Following Sec. 3.5, we can classify whether a point on the ridge-valley graph corresponds to a (pseudo-)ridge or a (pseudo-)valley by checking the signs of the 2nd-order derivatives of f and g in the direction tangent to the level set of f . The direction \mathbf{m} tangent to a contour of f is given by Eq. (11). The 2nd-order derivatives of f and g in the \mathbf{m} direction are $f_{\mathbf{m}\mathbf{m}} = \mathbf{m}^\top H \mathbf{m}$ and $g_{\mathbf{m}\mathbf{m}} = \mathbf{m}^\top H_g \mathbf{m}$ respectively. Tab. 1 summarizes the classification based on the signs of $f_{\mathbf{m}\mathbf{m}}$ and $g_{\mathbf{m}\mathbf{m}}$.

Table 1: Ridge-valley graph point classification based on the signs of $f_{\mathbf{m}\mathbf{m}}$ and $g_{\mathbf{m}\mathbf{m}}$.

Classification	$f_{\mathbf{m}\mathbf{m}}$	$g_{\mathbf{m}\mathbf{m}}$
Ridge point	< 0	> 0
Valley point	> 0	> 0
Pseudo-ridge point	< 0	< 0
Pseudo-valley point	> 0	< 0

4.4 Time Complexity

We work with 2D MFA models with degree-4 polynomials ($p = 4$). Let n denote the number of spans. Querying the function value and the gradient at a given point takes $\mathcal{O}(p^2)$. Using gradient descent to locate the starting points for particle tracing takes $\mathcal{O}(c_{\max} p^2)$, where c_{\max} is the predefined maximum iteration number of gradient descent. In each span, as we set $(p+3)^2$ initial points for gradient descent, finding all starting points takes $\mathcal{O}(c_{\max} p^4)$. Each application of RK4 computes four gradients and takes $\mathcal{O}(p^2)$. Assume we trace k points in a span, particle tracing takes $\mathcal{O}(kp^2)$. The overall time complexity for particle tracing in a span is $\mathcal{O}(kp^2 + c_{\max} p^4)$. The time complexity to locate all critical points is $\mathcal{O}(i_{\max} p^4 n)$, where i_{\max} is the maximum number of iterations using the Newton’s method [29]. In a single span, there are at most $(p+3)^2$ trajectories; and establishing the connection among different trajectories and critical points takes $\mathcal{O}(p^4)$ per span. In summary, the overall time complexity for contour extraction is $\mathcal{O}(nkp^2 + n(c_{\max} + i_{\max})p^4)$. As Jacobi set and ridge-valley graph extraction are based on contour extraction from derived functions, their time complexity stays the same.

5 EXPERIMENTAL RESULTS

To evaluate our framework, we perform a number of experiments with MFA models representing synthetic and scientific datasets; see the supplement for details on these MFA models. We address the following questions: Given an MFA model as input, is it possible to extract complex topological descriptors without discretizing the entire domain? Furthermore, how does our method perform in comparison to discrete methods that rely on model discretization?

For each MFA model, we extract contours, Jacobi sets, or ridge-valley graphs from the model, and compare the results against certain discrete methods.

Implementation. The code is compiled using g++ version 11.4.0 with the -O3 optimization flag. Multithreading is achieved through Threading Building Blocks (TBB), which dynamically balances the workload across threads. The number of threads is configured to match the number of hardware cores. All our experiments are conducted on a desktop powered by an Intel i9 CPU (3.5GHz) with 8 hardware cores and 8 threads, paired with 32 GB of DDR4 RAM.

A naive strategy for extracting topology from a continuous model is to sample it into a discrete representation and then apply established discrete methods. For comparative analysis, we discretize the MFA model onto a mesh to obtain a discrete representation. After triangulation, we use ParaView [4] to extract contours

and the Topology ToolKit (TTK) [55] to extract Jacobi sets and ridge-valley graphs from the resulting discrete representation. ParaView and TTK work with piecewise-linear (PL) interpolations of scalar functions defined over triangulated meshes. In particular, we use Contour and TTKJacobiSet filters to extract contours and Jacobi sets from discrete representations, respectively. The Contour filter in ParaView implements the marching squares algorithm [27] to extract the contour. We note that TTKJacobiSet typically includes the entire domain boundary in the resulting Jacobi set, even though these boundary elements may not genuinely belong to it; hence, we exclude domain boundaries from the Jacobi set comparisons. We use TTKJacobiSet to extract the ridge-valley graph as a special type of Jacobi set. We refer to our approach as the *MFA continuous method*, and refer to the methods based on discrete representations collectively as the *discrete method*.

Experimental setup. The goal of our experiments is to assess whether our MFA continuous method can effectively extract contours, Jacobi sets, and ridge-valley graphs from a continuous model, especially considering that the ground truth for these topological descriptors is typically unavailable. We compare our results to those obtained using the discrete method, which processes a discrete sample drawn from the same MFA model. The outputs from these discrete methods serve as a reference, not the ground truth, because these discrete methods operate on a PL approximation generated from a discretization of the MFA model, rather than on the MFA model itself. Despite these methodological differences, these discrete methods provide a meaningful benchmark for assessing how closely our results align with an established discrete pipeline. Fig. 4 illustrates an overview of contour extraction using both methods.

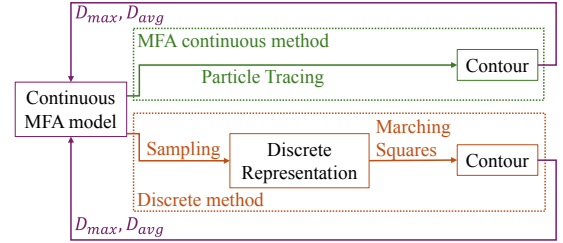


Figure 4: Overview of the MFA continuous method vs. the discrete method.

Our experimental procedure is summarized as follows:

1. Given an MFA model as the input, we apply our MFA continuous method to extract topological descriptors—contours, Jacobi sets, and ridge-valley graphs—from the model.
2. For comparative analysis, we generate a discrete representation by sampling points from the MFA model and apply ParaView or TTK to their PL interpolation to extract the corresponding topological descriptors.
3. We evaluate the results using metrics detailed below.

Evaluation metrics. Since both approaches are derived from the same MFA model, we can assess the quality of the contours extracted by each method by measuring the difference between the MFA function value $f(\mathbf{x})$ and the prescribed isovalue a at each point \mathbf{x} in the domain,

$$e(\mathbf{x}) = |f(\mathbf{x}) - a|. \quad (22)$$

For Jacobi set extraction, we reformulate the task into a 0-isovalue contour extraction of function h as shown in Eq. (15), resulting in the difference measure $e(x) = |h(x)|$. Similarly, for ridge-valley graph extraction, the discrepancy is computed as $e(\mathbf{x}) = |\tilde{h}(\mathbf{x})|$, where $\tilde{h}(\mathbf{x})$ is given by Eq. (21). We report both the maximum difference e_{\max} and average difference e_{avg} across all points x . To quantitatively evaluate the extracted topological descriptors, we report the number of loops (#Loop) and the number of connected components (#CC).

5.1 An Overview of MFA Models

1. Contour: Applied to **Sinc**, **S3D**, and Schwefel.
 2. Jacobi set: Applied to **Gaussian pair**, **von Kármán vortex street**, **Hurricane Isabel**, and Boussinesq approximation.
 3. Ridge-valley graph: Applied to **Gaussian mixture** and **CESM**.
- Here, experiments in bold are described in this section and others are included in the supplement.

We explore four synthetic MFA models (Schwefel, Sinc, Gaussian pair, and Gaussian mixture) and five scientific MFA models (S3D, von Kármán vortex street, Hurricane Isabel, Boussinesq approximation, and CESM); see the supplement for details. We use synthetic MFA models to validate the accuracy of our results, and scientific MFA models to demonstrate the efficacy of our framework. The computation time is listed in Tab. 2. With 8 threads, our method achieves a speedup of approximately $7\times$ to $8\times$, highlighting its parallel efficiency. Although our method is slower than the discrete approach, it avoids model discretization and enables direct analysis on the continuous model. This represents a fundamentally different strategy that offers improved fidelity for downstream topological analysis.

Table 2: Computation time in seconds.

MFA Models	MFA Continuous Method		Discrete Method
	Single Thread	8 Threads	Single Thread
Step Size $s = l/4$, Sampling Ratio 4			
Schwefel (Contour, $a = 100$)	8.97	1.24	0.0466
Sinc (Contour, $a = 0.33$)	1.49	0.201	0.0411
Gaussian Pair (Jacobi set)	1.61	0.233	0.0530
Gaussian Mixture (Ridge-valley graph)	50.6	6.43	0.310
Step Size $s = l/16$, Sampling Ratio 16			
S3D (Contour, $a = 50$)	17.7	2.37	5.71
Step Size $s = l/32$, Sampling Ratio 32			
Kármán (Jacobi set)	131	17.7	10.9
Boussinesq (Jacobi set)	557	76.6	23.2
CESM (Ridge-valley graph)	3218	408	115
Step Size $s = l/64$, Sampling Ratio 64			
Hurricane (Jacobi set)	6289	857	394

5.2 Sinc: Contour Extraction

Parameter selection via ablation studies. We select our parameters—step size s , accuracy threshold ϵ , and trajectory connection threshold γ —based on a series of ablation studies. Recall from Sec. 3.1 that the *span length* l is the distance in the MFA model between knots and corresponding control points. All the MFA models in this study have a uniform span length across the entire model. Table 3: Sinc model: evaluation of contour extraction with various isovalues (a). GT denotes the ground truth.

a	e_{\max}	e_{avg}	#Loop	GT#Loop	#CC	GT#CC
0.33	$9.8e^{-11}$	$5.5e^{-12}$	40	40	60	60
0.79	$9.9e^{-11}$	$5.1e^{-12}$	28	28	32	32

For all synthetic and scientific MFA models, we use a model-specific step size in each experiment, selected based on an ablation study. We decrease step size $s \in \{l/2, l/4, l/8, \dots, l/2^p, \dots\}$ until both #Loop and #CC reach convergence. The accuracy threshold is chosen at $\epsilon = 1e^{-10}$ for all experiments. Trajectories are connected when the distance between them is within a connection threshold $\gamma = 2s$ as described in Sec. 4.1.3; see the supplement for further details on parameter selection.

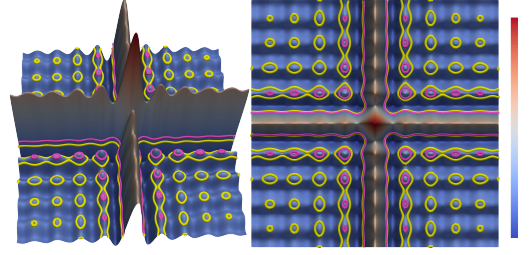


Figure 5: Sinc model: contour extraction with isovalues $a = 0.33$ (yellow) and $a = 0.79$ (pink). Results are shown in different views.

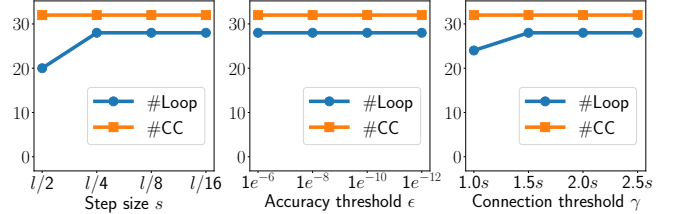


Figure 6: Contour extraction of Sinc model at $a = 0.79$. Number of loops and connected components vs. step size s (left), accuracy threshold ϵ (middle), and trajectory connection threshold γ (right).

When constructing a discrete representation from an MFA model, we set the grid spacing equal to the chosen step size s . This ensures that the edge lengths of the PL contour in the discrete representation remains consistent with those extracted from the continuous representation. For example, a grid spacing of $l/4$ results in sampling 4 points per dimension within each span, corresponding to a sampling ratio of 4.

In Fig. 5, contours are extracted from the Sinc model using isovalues $a = 0.33$ and $a = 0.79$. The original Sinc function serves as the ground truth. Tab. 3 shows that the errors at all nodes are below the threshold $\epsilon = 1e^{-10}$. The numbers of loops and connected components align exactly with those of the ground truth. In Fig. 6, we illustrate the results for varying step sizes (s), thresholds (ϵ), and connection thresholds (γ) at $a = 0.79$. For the Sinc model, step size $s = l/4$ is sufficient to ensure convergence for the number of connected components and loops. Under these conditions, the results demonstrate robustness with respect to variations in ϵ and γ .

5.3 S3D: Contour Extraction

Tab. 4 presents an evaluation of our MFA continuous method with a step size of $s = l/16$ alongside the discrete method with a sampling ratio of 16, using the original continuous MFA model as a reference. The discrete method differs primarily due to discretization artifacts. At $a = 50$, it misses a small loop that is visible in the black block of Fig. 7 (1) but absent in Fig. 7 (2). In Fig. 7, the three contours at isovalues $a = 30, 50, 60$ from the S3D model are displayed using both continuous and discrete representations.

In Fig. 8, results are visualized for different step sizes (s), thresholds (ϵ), and connection thresholds (γ) at $a = 50$. A step size of $s = l/16$ is chosen to guarantee convergence of both connected components and loops. This selection yields robust results with respect to variations in ϵ . Additionally, $\gamma = 2s$ is adopted for consistency across experiments and to ensure convergence.

Table 4: S3D model: contour extraction with step size $s = l/16$ and a sampling ratio 16.

a	MFA Continuous Method				Discrete Method			
	e_{\max}	e_{avg}	#Loop	#CC	e_{\max}	e_{avg}	#Loop	#CC
30	$1.0e^{-10}$	$5.6e^{-12}$	21	23	0.25	$8.5e^{-3}$	21	23
50	$1.0e^{-10}$	$4.5e^{-12}$	18	22	0.23	$1.1e^{-2}$	17	21
60	$1.0e^{-10}$	$1.0e^{-11}$	13	21	0.22	$6.4e^{-3}$	13	21

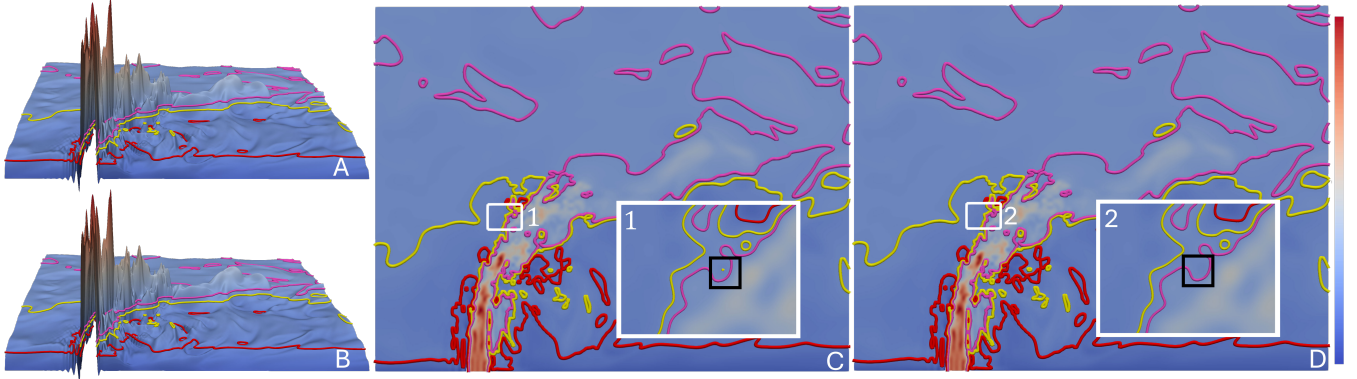


Figure 7: S3D model: contour extraction using isovalues $a = 30$ (red), $a = 50$ (yellow), and $a = 60$ (pink). (A, C): results from MFA continuous method with a step size of $s = l/16$, viewed from the top and the side; (B, D): results from discrete method with a sampling ratio of 16, viewed from the top and the side. (1) and (2) provide the zoomed-in views of the white blocks in (C) and (D), respectively.

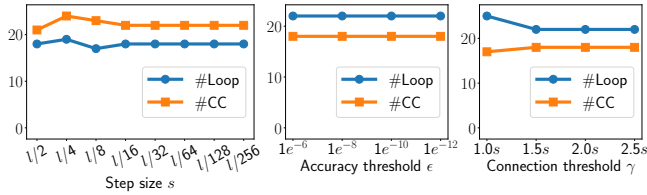


Figure 8: Contour extraction of S3D model at $a = 50$. Number of loops and connected components vs. step size s (left), accuracy threshold ϵ (middle), and connection threshold γ (right).

Table 5: Jacobi set extraction of MFA models.

MFA Continuous Method				Discrete Method			
e_{\max}	e_{avg}	#Loop	#CC	e_{\max}	e_{avg}	#Loop	#CC
Gaussian Pair, Step Size $s = l/4$, Sampling Ratio 4							
$6.8e^{-11}$	$4.3e^{-12}$	0	2	0.31	$2.2e^{-2}$	161	2
Kármán, Step Size $s = l/32$, Sampling Ratio 32							
$1.0e^{-10}$	$2.2e^{-11}$	82	124	1.3	$1.5e^{-2}$	5039	559
Hurricane, Step Size $s = l/64$, Sampling Ratio 64							
$1.0e^{-10}$	$2.9e^{-11}$	1901	2015	1.0	$4.2e^{-3}$	$4.1e^5$	1956

5.4 Gaussian Pair: Jacobi Set Extraction

Given a pair of scalar functions f and g represented by MFA models, Fig. 1 presents the Jacobi sets $\mathbb{J} := \mathbb{J}(f, g)$ extracted from both the continuous Gaussian pair model (C) and its discrete representation (F). The scalar functions f and g are depicted as purple and red contours, respectively. Results from discrete representations exhibit visible artifacts due to discretization: most notably, they exhibit numerous zigzag patterns, in comparison with significantly smoother results from the continuous MFA model.

Quantitative evaluation is presented in Tab. 5. Due to artifacts arising from discretizing a continuous model, the discrete method exhibits significantly larger error compared to the MFA continuous method. Specifically, zig-zag patterns result in numerous small triangles, causing the discrete method to generate many spurious loops.

5.5 Von Kármán Vortex Street: Jacobi Set Extraction

Fig. 9 presents a visualization of the von Kármán vortex street MFA model at time step 1500. The step size is $s = l/32$, and the sampling ratio is 32. Due to resolution limitations, the discrete representation (B) exhibits pronounced zigzag patterns, while the continuous representation (A) is significantly smoother. These differences

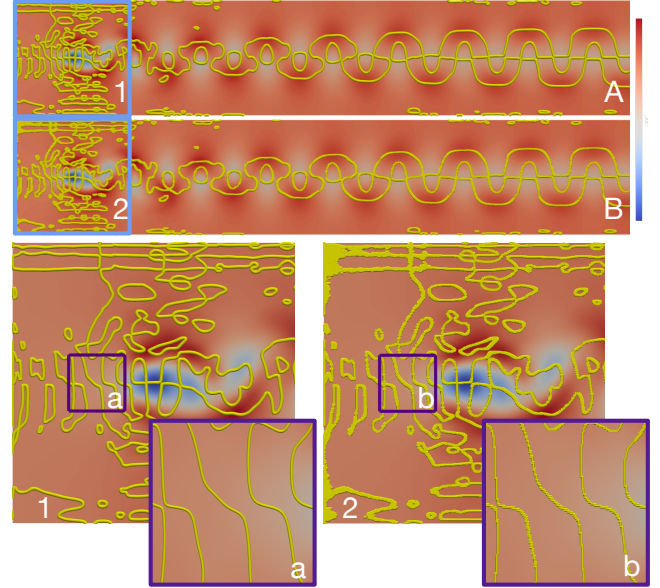


Figure 9: Von Kármán vortex street model: Jacobi set extraction using MFA continuous method (A) and discrete method (B). (1) and (2): zoomed-in views of blue blocks in (A) and (B), respectively. (a) and (b): zoomed-in views of purple blocks in (1) and (2), respectively.

are highlighted in zoomed-in views of (1) and (2), where zigzag patterns from the discrete representation are clearly visible in block (b) of (2), c.f., block (a) of (1). In Tab. 5, the error of the MFA continuous method remains below $\epsilon = 1e^{-10}$, whereas the error from the discrete method is much larger. The discrete representation generates many spurious loops due to these zig-zag patterns.

5.6 Hurricane Isabel: Jacobi Set Extraction

In Fig. 10, we present the Jacobi set extracted from the Hurricane Isabel model computed between temperature and pressure fields. Results are shown for the region $[125, 375] \times [120, 330]$. We select a slice of the model at a height of 50, where the hurricane exhibits significant spatial expansion and displays many characteristic structures [22]. The step size for MFA continuous method is $s = l/64$, and the corresponding sampling ratio is 64. The results from the continuous representation in blocks (1) and (2) are smoother than the results from the discrete representations in blocks (3) and (4), respectively. Zig-zag patterns, like those in the purple block of (4), contribute to the high number of loops in the discrete representation, as shown in Tab. 5.

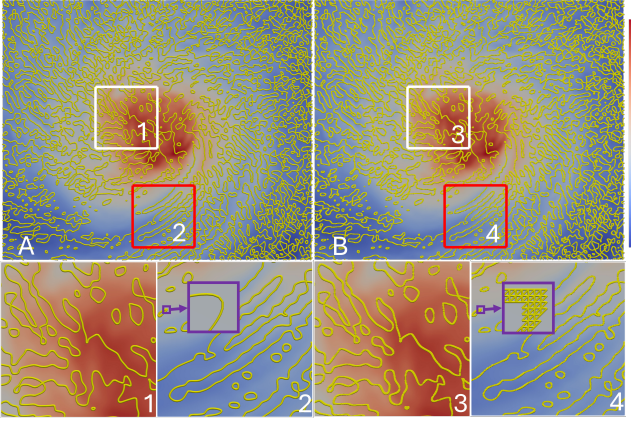


Figure 10: Hurricane Isabel model: Jacobi set extraction using MFA continuous method (A) and discrete method (B). (1-4): zoomed-in views of the white (1,3) and red (2,4) blocks, respectively. Purple insets further magnify the highlighted regions indicated by arrows.

Table 6: Ridge-valley graph extraction from MFA models.

MFA Continuous Method				Discrete Method			
e_{\max}	e_{avg}	#Loop	#CC	e_{\max}	e_{avg}	#Loop	#CC
Gaussian Mixture, Step Size $s = l/4$, Sampling Ratio 4							
$9.9e^{-11}$	$5.9e^{-12}$	2	1	1.4	$4.3e^{-1}$	970	1
CESM, Step Size $s = l/32$, Sampling Ratio 32							
$1.0e^{-10}$	$1.9e^{-11}$	3402	860	$4.2e^2$	$2.7e^2$	$2.1e^5$	751

5.7 Gaussian Mixture: Ridge-Valley Graph Extraction

Fig. 1 presents the results of ridge-valley graph extraction using the MFA continuous method (B) and the discrete method (E). Based on a convergence study, the step size for (B) is set to $s = l/4$, and the sampling ratio for (E) is 4. While the overall ridge-valley graph structures align for both methods, the discrete representation in (E) exhibits pronounced zigzag patterns and generates hundreds of spurious loops (see #Loop in Tab. 6). Moreover, the discrete method shows a significantly larger error compared to the original MFA model (see e_{\max} and e_{avg} in Tab. 6).

By utilizing the advantages of a continuous representation that allows access to second-order derivatives, our method enables the classification of any point on the ridge-valley graph based on its position; see Tab. 1. In Fig. 11, we present the classification of arcs from the extracted ridge-valley graph.

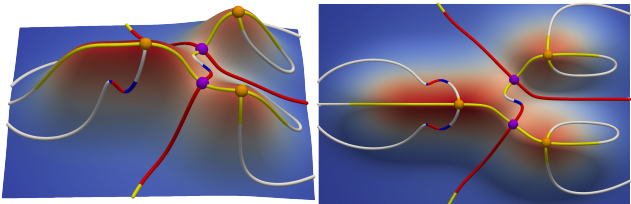


Figure 11: Gaussian mixture model: Ridge-valley graph extraction and classification with 3D (left) and 2D (right) views. Local maxima (orange) and saddles (purple) are connected by ridges (yellow), valleys (red), pseudo-ridges (white), or pseudo-valleys (blue).

5.8 CESM: Ridge-Valley Graph Extraction

Fig. 12 presents the results of ridge-valley graph extraction, with a step size $s = l/32$ for the MFA continuous method and a sampling ratio of 32 for the discrete method. Our method yields smoother results compared to the discrete representation, which introduces zigzag artifacts and numerous loops, as reported in Tab. 6. These



Figure 12: CESM model: Ridge-valley graph extraction for block $[2087, 2411] \times [1367, 1691]$ using MFA continuous method (left) and discrete method (right).

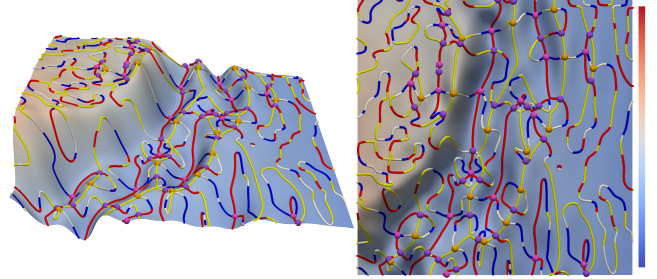


Figure 13: CESM model: Ridge-valley graph extraction and classification with 3D (left) and 2D (right) views. Local maxima (orange), saddles (purple), and local minima (pink) are connected by ridges (yellow), valleys (red), pseudo-ridges (white), or pseudo-valleys (blue).

discrepancies between the continuous model and its discretization contribute significantly to the error observed for the discrete method in Tab. 6. In Fig. 13, we present the classification of arcs from the extracted ridge-valley graph based on the MFA continuous method.

6 CONCLUSION AND DISCUSSION

We present a novel framework for extracting contours, Jacobi sets, and ridge-valley graphs directly from continuous implicit models that allow queries of function values and high-order derivatives. In our experiments, we focus on MFA models as examples of continuous implicit models. Our framework directly works on MFA models without requiring discretization, and leverages multithreading for enhanced performance. We demonstrate the effectiveness of our framework across various scientific datasets, showcasing its capability to support topological data analysis with continuous implicit models. In the future, we would like to extend our framework to extract other topological descriptors such as Morse/Morse–Smale complexes, which are similar but not quite the same as the ridge-valley graphs. Furthermore, extending these features to 3D domains is a promising and impactful direction for future research.

Limitations. During particle tracing, we identify multiple trajectories within each span that need to be connected. Our connecting strategy may introduce incorrect connections when trajectories are close enough, and the tracing process may produce undesirable gaps between trajectories. Reducing the step size until convergence of the number of connected components and loops will mitigate these issues, as shown by the ablation studies. Finally, the selection of initial points (for locating starting points during particle tracing) may affect the results. For certain MFA models, a greater number of initial points may be required within a span to ensure that starting points exist for every piece of the contours in that span. Developing a clear strategy with such guarantees is left for future work.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under contract numbers DE-AC02-06CH11357, DE-SC0023157, and DE-SC0022753, program manager Hal Finkel. It was also supported in part by National Science Foundation (NSF) grant DMS-2301361.

REFERENCES

- [1] HepOnHPC. <https://computing.fnal.gov/hep-on-hpc/>. Accessed: 2025-6-16. 1
- [2] RAPIDS2. <https://rapids.lbl.gov/>. Accessed: 2025-6-16. 1
- [3] Seahorse. <https://seahorse-scidac.github.io/>. Accessed: 2025-6-16. 1
- [4] J. Ahrens, B. Geveci, and C. Law. ParaView: An end-user tool for large data visualization. In *Visualization Handbook*. Elsevier, 2005. ISBN 978-0123875822. 6
- [5] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction*, vol. 13 of *Springer Series in Computational Mathematics*. Springer, 2012. doi: 10.1007/978-3-642-61257-2 2
- [6] H. Bhatia, B. Wang, G. Norgard, V. Pascucci, and P.-T. Bremer. Local, smooth, and consistent Jacobi set simplification. *Computational Geometry*, 48(4):311–332, 2015. doi: 10.1016/j.comgeo.2014.10.009 4
- [7] J. Bloomenthal and B. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann Publishers Inc., 1997. 2
- [8] H. Carr and D. Duke. Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1100–1113, 2014. doi: 10.1109/TVCG.2013.269 2
- [9] A. Chattopadhyay, H. Carr, D. Duke, and Z. Geng. Extracting Jacobi structures in Reeb spaces. *EuroVis - Short Papers*, 2014. doi: 10.2312/eurovisshort.20141156 2
- [10] J. Damon. Generic structure of two-dimensional images under Gaussian blurring. *SIAM Journal on Applied Mathematics*, 59(1):97–138, 1998. doi: 10.1137/S0036139997318032 2
- [11] C. De Boor. *A practical guide to splines*, vol. 27 of *Applied Mathematical Sciences*. Springer-Verlag New York, revised ed., 2001. 2, 3
- [12] T. Dokken. Finding intersections of b-spline represented geometries using recursive subdivision techniques. *Computer Aided Geometric Design*, 2(1):189–195, 1985. doi: 10.1016/0167-8396(85)90024-X 2
- [13] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *Journal of Mathematical Imaging and Vision*, 4:353–373, 1994. doi: 10.1007/BF01262402 2
- [14] H. Edelsbrunner and J. Harer. Jacobi sets of multiple Morse functions. *Foundations of Computational Mathematics, Minneapolis*, 8:35–57, 2002. doi: 10.1017/CBO9781139106962.003 2, 3, 4
- [15] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Local and global comparison of continuous functions. In *IEEE Visualization 2004*, pp. 275–280, 2004. doi: 10.1109/VISUAL.2004.68 5
- [16] A. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, and C. Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Publishing Company, Incorporated, 1st ed., 2009. 2
- [17] 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. *Computer Graphics Forum*, 35(3):643–667, 2016. doi: 10.1111/cgf.12933 1
- [18] L. Huettenberger and C. Garth. A comparison of Pareto sets and Jacobi sets. *Topological and Statistical Methods for Complex Data*, pp. 125–141, 2015. doi: 10.1007/978-3-662-44900-4.8 2
- [19] L. Huettenberger, C. Heine, H. Carr, G. Scheuermann, and C. Garth. Towards multifield scalar topology based on Pareto optimality. *Computer Graphics Forum*, 32:341–350, 2013. doi: 10.1111/cgf.12121 2
- [20] M. Jansen and P. Oninix. *Second generation wavelets and applications*. Springer, London, 2005. doi: 10.1007/1-84628-140-7 2
- [21] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*, pp. 339–346, 2002. doi: 10.1145/566570.566586 2
- [22] D. Klötzl, T. Krake, Y. Zhou, I. Hotz, B. Wang, and D. Weiskopf. Local bilinear computation of Jacobi sets. *The Visual Computer*, 38(9):3435–3448, Sept. 2022. doi: 10.1007/s00371-022-02557-4 2, 8
- [23] D. Klötzl, T. Krake, Y. Zhou, J. Stober, K. Schulte, I. Hotz, B. Wang, and D. Weiskopf. Reduced connectivity for local bilinear Jacobi sets. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pp. 39–48, 2022. doi: 10.1109/TopoInVis57755.2022.00011 2
- [24] D. Lenz, R. Yeh, V. Mahadevan, I. Grindeanu, and T. Peterka. Customizable adaptive regularization techniques for B-spline modeling. *Journal of Computational Science*, 71:102037, 2023. doi: 10.1016/j.jocs.2023.102037 2
- [25] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of marching cubes’ cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003. doi: 10.1080/10867651.2003.10487582 2
- [26] C. Lopez-Molina, G. V.-D. De Ulzurrun, J. M. Baetens, J. Van den Bulcke, and B. De Baets. Unsupervised ridge detection using second order anisotropic gaussian kernels. *Signal Processing*, 116:55–67, 2015. doi: 10.1016/j.sigpro.2015.03.024 3
- [27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, pp. 163–169. Association for Computing Machinery, New York, NY, USA, 1987. doi: 10.1145/37401.37422 2, 6
- [28] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. *Computer Graphics Forum*, 40(3):135–146, 2021. doi: 10.1111/cgf.14295 2
- [29] G. Ma, D. Lenz, T. Peterka, H. Guo, and B. Wang. Critical point extraction from multivariate functional approximation. *2024 IEEE Topological Data Analysis and Visualization (TopoInVis)*, pp. 12–22, Oct. 2024. doi: 10.1109/TopoInVis64104.2024.00006 1, 2, 3, 6
- [30] Z. Majdisova and V. Skala. Radial basis function approximations: comparison and applications. *Applied Mathematical Modelling*, 51:728–743, 2017. doi: 10.1016/j.apm.2017.07.033 1, 2
- [31] D. Meduri, M. Sharma, and V. Natarajan. Jacobi set simplification for tracking topological features in time-varying scalar fields. *The Visual Computer*, 40(7):4843–4855, 2024. doi: 10.1007/s00371-024-03484-2 2
- [32] E. Mello Rella, A. Chhatkuli, E. Konukoglu, and L. V. Gool. Neural vector fields for implicit surface representation and inference. *International Journal of Computer Vision*, 2024. doi: 10.1007/s11263-024-02251-z 2
- [33] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, Dec. 2021. doi: 10.1145/3503250 1
- [34] S. N and V. Natarajan. *Simplification of Jacobi Sets*, pp. 91–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-15014-2.8 4, 5
- [35] G. Nielson. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297, 2003. doi: 10.1109/TVCG.2003.1207437 2
- [36] M. Niemeyer and A. Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11448–11459, June 2021. doi: 10.1109/CVPR46437.2021.01129 1
- [37] G. Norgard and P.-T. Bremer. Ridge-valley graphs: Combinatorial ridge detection using Jacobi sets. *Computer Aided Geometric Design*, 30(6):597–608, 2013. Foundations of Topological Analysis. doi: 10.1016/j.cagd.2012.03.015 2, 3, 4, 5
- [38] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019. doi: 10.1109/CVPR.2019.00025 2

- [39] T. Peterka, Y. Nashed, I. Grindeanu, V. Mahadevan, R. Yeh, and D. Lenz. Multivariate functional approximation of scientific data. In *Situ Visualization for Computational Science*, pp. 375–397, 2022. doi: 10.1007/978-3-030-81627-8_17 1, 2
- [40] T. Peterka, Y. S. Nashed, I. Grindeanu, V. S. Mahadevan, R. Yeh, and X. Tricoche. Foundations of multivariate functional approximation for scientific data. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 61–71, 2018. doi: 10.1109/LDAV.2018.8739195 1, 2
- [41] T. Peterka, R. Ross, B. Nouanesengsy, T. Y. Lee, H. Shen, W. Kendall, and J. Huang. A study of parallel particle tracing for steady-state and time-varying flow fields. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pp. 580–591, 2011. doi: 10.1109/IPDPS.2011.62 3
- [42] L. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag, 2 ed., 1997. doi: 10.1007/978-3-642-59223-2 3
- [43] D. Pokrajac and R. Lazic. An efficient algorithm for high accuracy particle tracking in finite elements. *Advances in Water Resources*, 25(4):353–369, 2002. doi: 10.1016/S0309-1708(02)00012-X 3
- [44] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 ed., 2007. doi: 10.5555/1403886 3
- [45] R. Reisenhofer, J. Kiefer, and E. J. King. Shearlet-based detection of flame fronts. *Experiments in Fluids*, 57(41), 2016. doi: 10.1007/s00348-016-2128-6 4
- [46] R. Reisenhofer and E. J. King. Edge, ridge, and blob detection with symmetric molecules. *SIAM Journal on Imaging Sciences*, 12(4):1585–1626, 2019. doi: 10.1137/19M1240861 3
- [47] M. Sharma and V. Natarajan. Jacobi set driven search for flexible fiber surface extraction. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pp. 49–58, 2022. doi: 10.1109/TopoInVis57755.2022.00012 2
- [48] G.-S. Shokouh, B. Magnier, B. Xu, and P. Montesinos. Ridge detection by image filtering techniques: A review and an objective analysis. *Pattern Recognition and Image Analysis*, 31(3):551–570, 2021. doi: 10.1134/S1054661821030226 4
- [49] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 1, 2
- [50] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 1
- [51] J. Sun, D. Lenz, H. Yu, and T. Peterka. Scalable volume visualization for big scientific data modeled by functional approximation. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 905–914, 2023. doi: 10.1109/BigData59044.2023.10386434 1, 2
- [52] J. Sun, D. Lenz, H. Yu, and T. Peterka. MFA-DVR: Direct volume rendering of MFA models. *Journal of Visualization*, 27(1):109–126, Oct. 2024. doi: 10.1007/s12650-023-00946-y 2
- [53] H. Theisel, T. Weinkauff, H.-C. Hege, and H.-P. Seidel. Stream line and path line oriented topology for 2d time-dependent vector fields. In *IEEE Visualization 2004*, pp. 321–328, 2004. doi: 10.1109/VISUAL.2004.99 2
- [54] J. Tierny and H. Carr. Jacobi fiber surfaces for bivariate Reeb space computation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):960–969, 2017. doi: 10.1109/TVCG.2016.2599017 2
- [55] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, Jan. 2018. doi: 10.1109/TVCG.2017.2743938 6
- [56] H. Wendland. *Scattered Data Approximation*, vol. 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539 2
- [57] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021. doi: 10.1111/cg.14331 1
- [58] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems*, 35:25018–25032, 2022. 2