



Topological Simplification of Jacobi Sets for Piecewise-Linear Bivariate 2D Scalar Fields with Adjustment of the Underlying Data

Felix Raith *
Leipzig University

Gerik Scheuermann †
Leipzig University & ScaDS.AI

Christian Heine ‡
Leipzig University

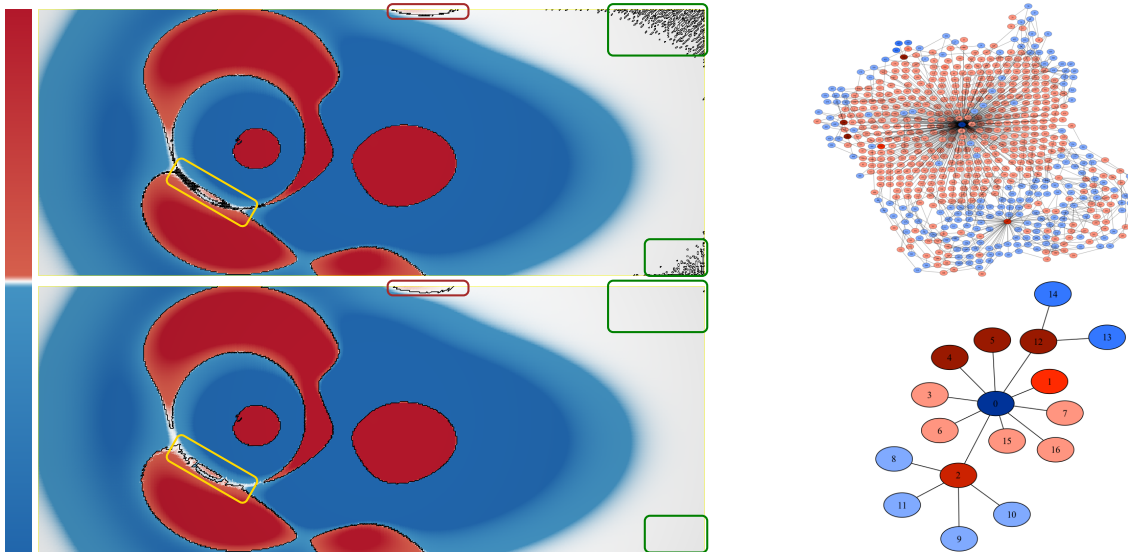


Figure 1: Comparison of the calculated Jacobi sets in the Cylinder Flow dataset on the left side of the figure for the original dataset in the upper figure before simplification and the dataset in the lower figure after simplification with the collapse algorithm variant A with threshold value $\iota = 0.0001$. Furthermore, the corresponding neighborhood graphs are shown on the right side, with the color corresponding to the orientation and the range area.

ABSTRACT

Jacobi sets are an important tool to study the relationship between functions. Defined as the set of all points where the function's gradients are linearly dependent, Jacobi sets extend the notion of critical point to multifields. In practice, Jacobi sets for piecewise-linear approximations of smooth functions can become very complex and large due to noise and numerical errors. Existing methods that simplify Jacobi sets exist, but either do not address how the functions' values have to change in order to have simpler Jacobi sets or remain purely theoretical. In this paper, we present a method that modifies 2D bivariate scalar fields such that Jacobi set components that are due to noise are removed, while preserving the essential structures of the fields. The method uses the Jacobi set to decompose the domain, stores the and weighs the resulting regions in a neighborhood graph, which is then used to determine which regions to join by collapsing the image of the region's cells. We investigate the influence of different tie-breaks when building the neighborhood graphs and the treatment of collapsed cells. We apply our algorithm to a range of datasets, both analytical and real-world and compare its performance to simple data smoothing.

Index Terms: Topological data analysis, bivariate data, Jacobi set, topological simplification.

*e-mail: raith@informatik.uni-leipzig.de

†e-mail: scheuer@informatik.uni-leipzig.de

‡e-mail: heine@informatik.uni-leipzig.de

1 INTRODUCTION

The analysis of multivariate data is frequently carried out in science, and the analysis of bivariate data in particular has established itself as an important field, for example in climate simulations where temperature and pressure are jointly investigated. The relationship between the scalar fields can be examined using Jacobi sets [18] as a tool for analysis. This mathematical concept originates from the field of topology and represents a generalization of critical points to multifields. In topological data analysis, Jacobi sets are used to extract Ridge-Valley graph in computer vision and image processing [31], and to track critical points in time [4, 18]. Besides, they are used to compare scalar fields [19] to study the Reeb space [39] and to drive fiber surface extraction [34]. They are not only used in visualization but also in other scientific disciplines [1, 2, 25, 30]. This multifaceted application shows the importance of Jacobi sets in scientific analysis, which is why these structures must be easy to understand.

However, noise can be a major problem that makes analysis difficult for domain experts. For this reason, a work on the simplification of Jacobi sets [37] has already been carried out, the Reeb graphs created and the Jacobi sets simplified with the comparison measure κ . This involves removing loops or Jacobi set components, which occur in particular in the case of noise and numerical errors. However, only the representation of the Jacobi sets is simplified, the underlying data itself is not adjusted, which is unfavorable for further processing. In contrast, smoothing filters adjust the data globally, but the extent to which important structures are lost in the process has not been sufficiently investigated. Important structures are regions in the dataset that lie above a threshold value of a comparison metric and are also visually separated from the surrounding area, see Fig. 1 in the upper

left part in the brown region of interest. Subdivision algorithms can also simplify Jacobi sets and, in particular, remove zig-zag, but only lead to a better representation of this [27]. However, the visual evaluation indicates that the number of Jacobi set components of the dataset increases. Therefore, we will compare these approaches and our approach to determine how suitable they are for simplifying the Jacobi sets.

Motivated by the theoretical approach of Bhatia et al. [3], we have developed a method that aims to change the functions by *collapsing* a cell to reduce the complexity of the Jacobi sets while retaining their important structures by changing the underlying data. The *collapse* of a cell means that the area that this cell sets up in the range is reduced to the value 0, which is also the case with degenerated cells. This approach simplifies the Jacobi sets and thus also reduces the Jacobi set components. The idea of collapsing comes from the 1D case, in which 2 critical points can be reduced by mapping the function values $f(x)$ of the two points of a cell to the same value. In Fig. 3 this is illustrated for the cell BC . Here you can see that this change affects at least one neighboring cell. Therefore, the choice of a suitable value is also crucial to keep the side effects as low as possible, as otherwise not all critical points can be removed. In the 2D case, this concept is more difficult, as the number of affected neighboring cells increases. Therefore, a suitable metric must be used to keep the side effects as low as possible. To identify the components to be collapsed, we use the neighborhood graph of the Jacobi set components.

The contributions in this paper are:

- We present an algorithm for simplifying Jacobi sets based on iteratively collapsing cells while changing the underlying data and identifying this with the help of a neighborhood graph.
- We investigate the influence of different neighborhood graphs on this algorithm.
- We present an adapted Jacobi sets visualization that assigns degenerate cells according to their point neighborhood.
- We demonstrate the effect of smoothing and loop subdivision on the simplification of Jacobi sets using analytical and real datasets and compare them with our algorithm.

2 RELATED WORK

An overview of topology-based methods in visualization can be found in the survey by Heine et al. [21]. He et al. [20] focuses more on the visualization of multivariate data.

In scalar field topology, the essential features of scalar fields can be described by contour trees [8, 10, 42], Reeb graphs [33] or the Morse-Smale complex [16]. With their method, scalar fields can be topologically simplified and the critical points and their relationships can be reduced. Carr et al. [9] describe a method for simplifying the contour tree by suppressing smaller topological features. Bremer et al. [5] create the Morse-Smale complex of a 2D function and simplify the topology by canceling pairs of critical points. Luo et al. [29] simplify critical points based on a point cloud. Edelsbrunner et al. [17] introduce the idea of persistent homology for the topological simplification of a point cloud, which was introduced by Cohen-Steiner et al. [14]. Edelsbrunner et al. [17] introduce the idea of persistent homology for the topological simplification of a point cloud, which extended Cohen-Steiner et al. [14] to persistence diagrams.

In order to apply the methods mentioned to multifield data, an adaptation is necessary. Methods such as Jacobi sets and Reeb graphs cannot be extended so easily. Besides, Carlsson et al. [6] showed that a generalization of persistent homology is difficult. Singh et al. [35] use partial clustering of high-dimensional data and introduce the idea of the mapper based on this. The Joint Contour Net by Carr

et al. [7] is partly based on this idea, but use joint contour plates and their topological connectivity. Reeb graphs were introduced in the works [13] and [36] parallel for multifields. Chattopadhyay et al. [13] introduce the Jacobi structure for subdividing the Reeb space, which creates a Reeb skeleton that corresponds to the Reeb graph. An algorithm for calculating the Reeb space of a bivariate, piecewise-linear scalar function on a tetrahedral grid is presented by Tierny et al. [38]. Chattopadhyay et al. [12] use the Joint Contour Net to further simplify multivariate data and to simplify the Reeb skeleton.

The simplification of Jacobi sets of multifield data has not seen much attention so far. The work by Bremer et al. [4] describes a method that removes noise from the Jacobi sets of time-varying data. In the work of Hüttenberger et al. [23, 24], a method for extending topological structures from single scalar fields to multifields using Pareto sets is proposed and the approach for simplifying Jacobi sets [37] is extended to multivariate data. Suthambhara and Natarajan [37] use reeb graphs to simplify the calculated Jacobi sets. Bhatia et al. [3] presented a theoretical approach to simplify Jacobi sets. In contrast to these works, our method also aims to simplify the underlying data with a simple approach.

Another approach to simplify the data is to use non-topological methods based on smoothing. This allows noise to be removed from the entire data, but this is heavily dependent on the filters used. Well-known filters are the binomial and Gaussian filters. Tong et al. [40] decompose the field into 3 parts, smooth them individually, and then sum them up again. In contrast, our method only processes the data in areas where noise occurs, which prevents other structures from being altered.

A completely different approach is presented by Klötzel et al. [27], they introduce a new method for calculating the Jacobi sets based on local bilinear interpolation, which implements a generalization of the definition by Edelsbrunner and Harer [18]. This method smoothes the Jacobi sets by reducing zig-zag patterns and better-resolving structures. This effect shows better results, especially at high resolutions. However, this leads to many small Jacobi set components in very noisy areas of the dataset.

3 BACKGROUND

A *scalar field* is a function $f : \mathbb{D} \rightarrow \mathbb{C}$, smoothly mapping from a *domain* \mathbb{D} , which is a compact d -manifold, to a *range* $\mathbb{C} \subset \mathbb{R}$. In this paper, we only consider domains that are subsets of \mathbb{R}^2 . A *critical point* \mathbf{x} of f is a point $\mathbf{x} \in \mathbb{D}$, where the gradient, i.e. the vector of partial derivatives, of f vanishes at \mathbf{x} : $\nabla f(\mathbf{x}) = \mathbf{0}$. A bivariate scalar field can be viewed as two scalar fields f, g mapping from the same domain. Edelsbrunner and Harer [18] introduced the *Jacobi set* $J(f, g)$ for a pair of functions f, g as the set of points $\mathbf{x} \in \mathbb{D}$, where the gradients $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{y})$ are linearly dependent:

$$J(f, g) := \{\mathbf{x} \in \mathbb{D} \mid \exists \lambda \in \mathbb{R} : \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = \mathbf{0} \text{ or } \lambda \nabla f(\mathbf{x}) + \nabla g(\mathbf{x}) = \mathbf{0}\} \quad (1)$$

Note that the critical points of f and g are trivially part of their Jacobi set. Edelsbrunner and Harer [18] showed that, if D is a 2-manifold, $J(g, f)$ is generically a collection of pairwise disjoint smooth curves, free of any self-intersections. We will call the connected components of a Jacobi set the *Jacobi set components*. Eq. 1 has many equivalent formulations, for example, Edelsbrunner et al. [19] mention that for two 2D functions, the Jacobi set is the set of all points where the rank of the Jacobian, i.e., the matrix comprised of the function's gradients, is smaller than 2. They also present a more general criterion: the set of all points x where the local κ measure is 0, where $\kappa(x)$ is defined as the length of the wedge product of the functions gradients. They argue that the average of this measure over an area is related to the amount of effort needed to change the topology of the functions in that area.

A k -simplex σ is the convex hull of $k + 1$ affinely independent points $P_\sigma = \{\mathbf{x}_0, \dots, \mathbf{x}_k\}$. 2-simplices are triangles, 1-simplices are line segments and 0-simplices are points. A simplex τ is a *face* of a simplex σ if $P_\tau \subseteq P_\sigma$. A *simplicial complex* is a set of simplices that are closed under the face relation and any two simplices' intersection is either empty or a common face. A *piecewise-linear* function assigns a function value to each 0-simplex and extends this to the other simplices using barycentric interpolation. Edelsbrunner and Harer [18] presented an algorithm to compute the Jacobi set for piecewise-linear functions approximating smooth functions, which we will not repeat here because our algorithm works slightly simpler, due to a more restricted setting.

4 SIMPLIFICATION OF BIVARIATE 2D SCALAR FIELDS

Our algorithm assumes that the data are given as two piecewise linear functions f, g on a compact domain $\mathbb{D} \subseteq \mathbb{R}^2$, which we will treat as a vector-valued function $\mathbf{f} : \mathbb{D} \rightarrow \mathbb{R}^2$. It first computes a measure for each triangle and uses it to determine the Jacobi set. It then uses the Jacobi set to decompose the domain into a set of regions separated by Jacobi set components and determines a neighborhood graph where the nodes represent the regions and are connected by an edge if separated by the same Jacobi set. It then assigns a weight to each region by suitably aggregating the measures from the first step over all triangles of this region. The weights are used to determine which region's functions values are to be manipulated in order to remove a Jacobi set component.

4.1 Jacobi Set Computation

If f, g were smooth, then due to fundamental theorems in linear algebra that relate the determinant of a matrix to the linear dependency of its rows and columns, Eq. 1 is equivalent to:

$$J(\mathbf{f}) = \{\mathbf{x} \in \mathbb{D} \mid \det \nabla \mathbf{f}(\mathbf{x}) = 0\}. \quad (2)$$

Note that, $\nabla \mathbf{f}(\mathbf{x})$ is called the *Jacobian* of \mathbf{f} in calculus, and is a matrix composed of the gradients of f and g . The absolute value of the Jacobian's determinant would equal the value of κ by Edelsbrunner et al. [19], but by not taking the absolute value, we get an alternative method to compute the Jacobi set for piecewise-linear representations of bivariate 2D fields. For piecewise-linear data the function restricted to the interior of each triangle σ can be given as:

$$\mathbf{f}|_\sigma(\mathbf{x}) = \mathbf{A}_\sigma \mathbf{x} + \mathbf{b}_\sigma. \quad (3)$$

If one uses a Taylor expansion of $\mathbf{f}|_\sigma$ at $\mathbf{x}_0 = 0$, one will find that $\mathbf{A}_\sigma = \nabla \mathbf{f}|_\sigma(\mathbf{x})$, and hence that the determinant of the Jacobian is constant across the triangle, but is typically discontinuous between neighboring triangles. Nevertheless, some cells will have a positive and some a negative determinant, and we determine the Jacobi set as the list of edges which are faces to triangles with different signs of the determinant.

Eq. 3 also gives a more geometrical intuition useful to understand the mechanics of our simplification algorithm. Essentially, Eq. 3 can be interpreted as a linear transformation of a space, i.e., how \mathbf{f} maps the triangle from the domain to the range. Due to this transformation triangles might get translated, rotated, scaled, and mirrored. If the determinant of \mathbf{A}_σ , i.e. the Jacobian, is negative, the triangle's image is mirrored, i.e., the order of the triangle's vertices switches between clockwise and counter-clockwise. If the determinant is 0 then the transformation will *collapse* the triangle, i.e., its image under \mathbf{f} is a line segment or a point. We will refer to the sign of the determinant of the Jacobian as the *orientation* of a triangle. The magnitude of the determinant furthermore gives the factor by which the area of the triangle grows or shrinks, indicating how much effort it takes to change the function values at a triangle's vertices to collapse it.

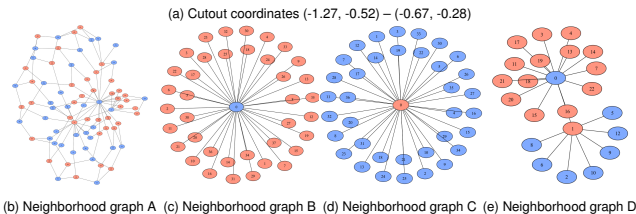
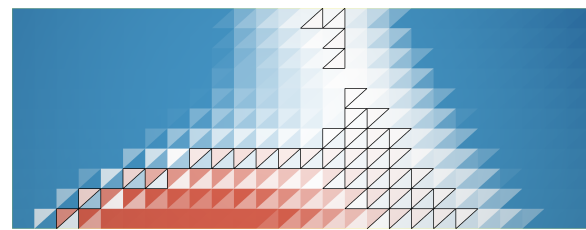


Figure 2: Section of the cylinder flow dataset (a) is an example of applying the different neighborhood graphs. (b) classical neighborhood graph A, (c) neighborhood graph B with connected negative cells, (d) neighborhood graph C with connected positive cells, and (e) neighborhood graph A with connected cells corresponding to the neighborhood.

4.2 Neighborhood graph

A neighborhood graph represents the relationship between objects through nodes and edges. A node represents a Jacobi set component, and an edge represents the geometric spatial proximity in the domain. We examine four approaches for creating neighborhood graphs labeled A to D, which differ in the grouping of the cells to the components. An example of the different of neighborhood graphs is shown in Fig. 2. This figure shows a section of a domain that is examined in more detail in Fig. 2a and illustrates all four approaches. The coloring of the cells represents two properties in the range: the orientation, where positive oriented triangles are colored red and negative oriented triangles are colored blue, and the area, where high saturation means a large area. The Jacobi sets are shown in black.

For the neighborhood graph A, all cells are combined into one node if they are not separated by a Jacobi set. This results in a graph with 69 nodes and 108 edges for the domain section under consideration. In neighborhood graph B, as shown in Fig. 2c, cells are merged into one node if they fulfill the conditions of graph A. Besides, two cells are combined into one node if they are connected via a point and both have a negative orientation. The neighborhood graph C, shown in Fig. 2d, follows the same approach as graph B, but here two cells are combined into one node if they are connected by a point and both have a positive orientation. These adaptations in graphs B and C are based on the goal of converting the neighborhood graph into a neighborhood tree. Such a tree can be simplified by starting to combine nodes at the leaves as long as the range area of these nodes are smaller than a threshold according to a suitable metric. A side effect of this adaptation is the reduction of nodes in the graph. The effects of these adaptations can be seen in the resulting graphs. The neighborhood graph B has 39 nodes and 38 edges, while the graph C has 33 nodes and 32 edges. The different orientations lead to significantly different graphs.

Finally, we consider the neighborhood graph D. Here, all cells are combined into one node if they are not separated by a Jacobi set. Besides, two cells are connected if they are connected by a point, have the same orientation and the summed orientation of all point neighbors is also the same. The goal of this approach is to minimize the imbalance that arises in the neighborhood graphs B and C and produce a more analyzable neighborhood tree. The resulting graph for this approach is shown in Fig. 2e and has 23 nodes and 22 edges. This graph shows a compact structure that may be well-suited for

further analysis.

To investigate the effect of the different neighborhood graphs on the simplification of the Jacobi sets, all four approaches are combined with the method developed, and the results are compared visually.

4.3 Metrics to select Jacobi set components

In this study, the goal is to collapse a selection of Jacobi set components to simplify the data. A suitable metric has to be chosen to select a Jacobi set component. There are geometric and topological metrics, and we have chosen a topological metric in this work. Based on the neighborhood graph, we now decide which of the Jacobi set components should be collapsed. Different measures can be used to choose the components:

- The Jacobi set component's domain area:

$$A_{\mathbb{D}} = \sum_{i=0}^n A_i \quad (4)$$

- The Jacobi set components range area:

$$A'_{\mathbb{C}} = \sum_{i=0}^n A'_i \quad (5)$$

- The hypervolume, which is the product of the Jacobi set components domain and range area:

$$HV = \sum_{i=0}^n A_i \cdot A'_i \quad (6)$$

Here i is a cell in a region. We chose hypervolume, which takes into account both the area of Jacobi set components in the domain and the range. Furthermore, the other two metrics pose problems, especially if the Jacobi set components areas are very different. This could lead to important structures being lost. Another possibility would be the automatic determination of this variable depending on the neighborhood. However, this method was not implemented in this work.

4.4 Collapse of Regions

To collapse a region, we use a greedy algorithm that shrinks the regions starting from the cells at the Jacobi set and collapses them first until the entire region is collapsed. The collapse a cell in the 2D case is a complex problem due to its influence on the neighborhood. In the 1D case, all values at the vertices of a cell can simply be set to the same value in the range, see Fig. 3. This collapses the line cell to a point in the 1D case. In the 2D case, Fig. 4a shows that V1 maps all values at the vertices of a triangle cell to one value and thus collapses the cell in the range. in Fig. 4a V2 collapsed the cell to a line in the range. The difference between these two variants is that collapsing to a point affects more neighboring cells compared to collapsing to a line. Therefore, we prefer collapsing to a line in this work. Since only one edge of the triangle cell needs to collapse to collapse the entire cell, we decide which of the edges to select depending on the neighborhood. In this way, side effects can be reduced, and places where several small Jacobi set components will be removed are collapsed from the outside. Since we know at this point which cells should be collapsed, we check whether there are cells in the neighborhood that should also be collapsed, which we can then divide into four cases:

1. No cell to be collapsed in the neighborhood Fig. 4a,
2. One cell to be collapsed in the neighborhood Fig. 4b,

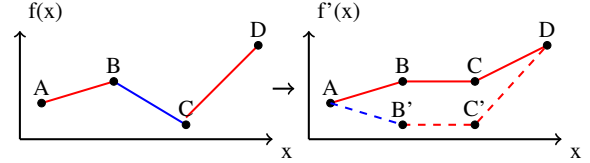


Figure 3: Example of collapsing a cell in 1D. Left before collapsing the cell BC, right after collapsing. The dashed line shows when the neighborhood is not taken into account. The solid line shows when it is taken into account. The red color shows a positive and the blue a negative orientation.

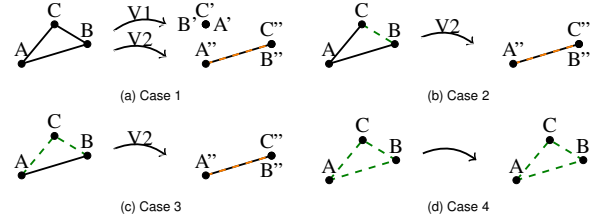


Figure 4: Example for the collapse of a triangle cell in the range for all 4 cases to be considered: V1 shows the collapse to a point, and V2 shows the collapse to a line. The green dashed edges connect the cell with a cell to be collapsed. The orange dashed lines are the edges to be scrambled.

3. Two cells to be collapsed in the neighborhood Fig. 4c,
4. Three cells to be collapsed in the neighborhood Fig. 4d.

In the first case, all edges of the triangle can be selected for collapsing. In the second case, the edges that connect the cell to a cell that is not to be collapsed should be retained. In the third case, only the edges that connect the cell to a cell to be collapsed can be collapsed. We skip the fourth case, as the collapsing of the neighboring cells means that after several iterations they are also on the edge of the Jacobi set component and can then be treated according to one of the other cases.

4.5 Method

The entire method is presented as pseudocode in Algorithm 1. Our algorithm uses a 2D bivariate scalar field defined on an unstructured grid. For these, a neighborhood graph is created in `ComputeNeighborhoodGraph(F)`. In this graph, the nodes represent individual Jacobi set components, while the edges represent the relationships between the different components. There are various ways in which the Jacobi set components can be composed in the neighborhood graph. These ways and their specifics are described in detail in Sec. 4.2.

This neighborhood graph is used in our method to select Jacobi set components. This is done in `FindCollapsibleCells(NG, \tau)`, which uses the hypervolume from Sec. 4.3 and an appropriate threshold to identify nodes that are considered irrelevant. The algorithm is cell-based and therefore returns all cells in a list (CL) that belong to the irrelevant nodes in a list. The algorithm now begins to process the list CL. For each cell, it calculates whether it lies on the border of a Jacobi set component to be collapsed. To do this, `CalculateCellNeighborhood(CL, c)` returns for a given cell how many neighboring cells are connected to it and are not contained in CL. This number is used to sort cells and skip cells that are not on the border. For the remaining cells, the list (CV) is returned that contains all possibility variants of how the cell can be collapsed. These variant are described in more detail in Sec. 4.4. This list is now used to select the best variant for collapsing the

Algorithm 1: Collapse Algorithm

Input : 2D Bivariate Scalar Field F ,
Threshold t
Output : Updated 2D Bivariate Scalar Field F ,
Neighborhood Graph NG

```
1 begin
2    $NG := \text{ComputeNeighborhoodGraph}(F)$ 
3    $CL := \text{FindCollapsibleCells}(NG, t)$ 
4   while  $CL \neq \emptyset$  do
5     for  $c \in CL$  do
6        $nc := \text{CalculateCellNeighborhood}(CL, c)$ 
7       if  $nc = 0$  then
8         continue
9        $CV := \text{PossibleCollapseVariants}(nc)$ 
10       $bcv := \text{FindBestCollapseVariant}(F, CV, c)$ 
11       $\text{ApplyCollapseVariant}(F, c, bcv)$ 
12
13       $CL := CL \setminus \{c\}$ 
14       $CL := CL \cup \text{FlippedCellNeighbors}(NG, c)$ 
15    if  $\text{CellsOscillated}(CL)$  then
16      break
```

cell. For this purpose, a metric is used that decides which variant has the least side effects based on the direct point neighborhood of the cell. This metric counts how many neighboring cells are negatively affected when the different options are applied. A negative influence occurs when cells change their orientation in the range, as shown by the dashed line in Fig. 3 for the 1D case. Besides, the area in the range in the neighborhood is also taken into account, and changes are preferred if the total area in the range becomes smaller. After selecting the best variant, the bivariate scalar field in $\text{ApplyCollapseVariant}(F, c, bcv)$ is adjusted. This means that the values of the cell vertices are set to the same value according to the cases from Sec. 4.4. The cell is then removed from the list CL . Since the algorithm cannot always select a variant in which no side effects occur, $\text{FlippedCellNeighbors}(NG, c)$ recognizes negatively influenced cells and then adds them to the list CL .

To prevent the algorithm from oscillating, the $\text{CellsOscillated}(CL)$ function checks this and terminates the algorithm in this case. The result of this method is a modified bivariate scalar field whose neighborhood graph is greatly reduced and the Jacobi sets are also simplified.

4.6 Jacobi set Visualization

The calculation of Jacobi sets, which separate cells according to their range, poses a challenge when a cell is degenerate or collapsed, because a degenerate cell cannot be assigned to one of the two area orientations to be separated. A possible solution to this problem is to assign degenerate cells based on their neighborhood.

The Jacobi sets are calculated as described in Sec. 4.1. Besides, degenerate cells are assigned based on their point neighborhood of the Jacobi set components that predominate in the neighborhood. This means that the cell is assigned to the orientation that is larger when the neighboring cell orientations are summed up. If there are only degenerated cells in the direct neighborhood, the neighbors of the degenerated cells are considered recursively until a clear assignment is possible.

When applying this method to the four possible neighborhood graphs from Sec. 4.2, this leads to slightly different results when dealing with degenerated cells. For the neighborhood graph A and the neighborhood graph D, which are mapped in Fig. 2b, 2e, collapsed cells are assigned to the Jacobi set component as described.

In neighborhood graph B, shown in Fig. 2c, degenerated cells are assigned to a negative component as long as a negative cell exists in their neighborhood regardless of what the summed neighbor cell orientation is. For neighborhood graph C, which is mapped in Fig. 2d, the procedure is identical to that for the neighborhood graph B, except that it applies to positive cells.

These adjustments ensure a clear assignment of degenerated cells, which is important for further analysis and interpretation of the Jacobi sets. However, it cannot be guaranteed that the length of the Jacobi set is always the shortest.

5 RESULTS

To evaluate how much smoothing filter and loop subdivision as well as the developed collapse algorithm (CA) in all 4 variants simplify the Jacobi sets, we apply these algorithms to three datasets and compare them with the original data. To do this, we compare visually and with two measures whose results for all datasets in Tab. 1. These measures are the number of Jacobi set components, and the length of all Jacobi sets in the domain. The number of Jacobi set components is a good measure to investigate a simplification since this quantity can be derived directly from the neighborhood graph and can be considered both visually and as a quantity. However, this quantity alone could lead to misinterpretations if many Jacobi sets are connected to a large closed edge, which reduces the components but has a negative effect on the length of the Jacobi sets. Therefore, we also consider the length of all Jacobi sets as a second parameter to evaluate to what extent the structure could also be simplified.

In all figures, the color in the domain shows the orientation of the cell in the range, while at the same time, the range area is visualized via saturation. The demonstrations were run on a MacBook Pro (14-inch, 2021) with a Apple M1 Max, and 64 GB Ram. Implemented is the algorithm inside our framework with C++.

5.1 Simple Smoothing Approaches

For our evaluations, we use known smoothing methods on the one hand and loop subdivision on the other. These are classic smoothing filters to reduce noise and remove irregularities. More precisely, we use the Gaussian filter and the binomial filter. Loop subdivision is a method from computer graphics for smooth subdivision of triangular grids. This method, developed by Loop [28], has become a fundamental tool in computer graphics. For all comparisons we use $\sigma = 1000$ for the Gaussian filter, $r = 1$ for the binomial filter, and 4 subdivision steps in the loop subdivision. In the appendix, there are further comparisons with other σ settings for the Gaussian filter.

5.2 Cylinder Flow (Synthetic)

First, we consider the analytical *Cylinder Flow* [26] dataset, which is freely available on the ETH Zürich website [11]. The dataset is a regular 2D grid with a resolution of 450×200 and 500 time steps, on which a synthetic vector field is defined. This vector field was used as a co-gradient to a stream function of Jung et al. [26] and represents a simplified model of a Kármán vortex street. We use time step 1 for the comparisons and triangulate the grid.

The aim of the analysis of this dataset is to show the effect of the CA in all variants, the smoothing filters and the loop subdivision on the simplification of the Jacobi sets and to find out which CA variant has the greatest effect and to use this for the other datasets. The original data serves as a reference. The resulting datasets and the corresponding Jacobi sets are shown in Fig. 1 and Fig. 5.

When observing the datasets, three regions of interest (ROI) can be identified that are relevant for the comparison because they are noisier or the structures are complex. Nevertheless, the rest of the dataset is not uninteresting as important structures should not or only minimally be changed. The first ROI is marked green in the datasets and contains the two right corners where many small Jacobi

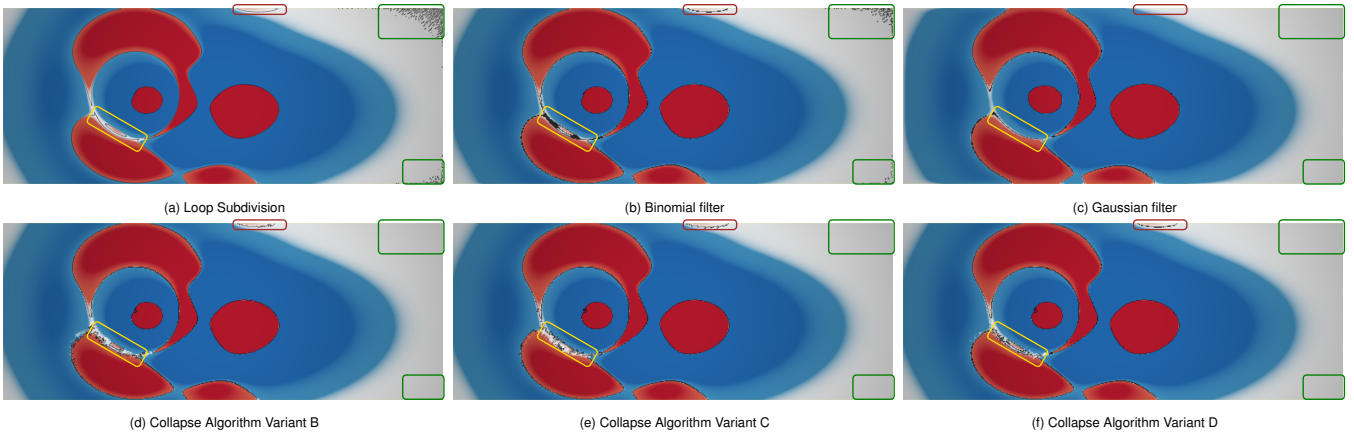


Figure 5: Comparison of the calculated Jacobi sets in the Cylinder Flow dataset for the Loop Subdivision (a), the Binomial filter (b), the Gaussian filter (c), the Collapse Algorithm variant B with $\tau = 0.0001$ (d), Collapse Algorithm variant C with $\tau = 0.0001$ (e), and Collapse Algorithm variant D with $\tau = 0.0001$ (f). Three regions of interest (ROI) are highlighted in color for visual comparison.

set components can be recognized in Fig. 1 of the original data. These are due to noise, as the range area is very small, which can also be seen from the saturation of the colors. In the loop subdivision, the Jacobi set components can still be recognized very well; a simplification of the Jacobi sets is therefore not possible. After smoothing with the binomial filter, a reduction of the Jacobi set components can already be recognized. The smoothing is further intensified with the Gaussian filter, which even leads to the removal of all Jacobi set components in this ROI. A similar result is obtained after applying the CA variants. Here, all Jacobi set components can also be removed, which can be seen in Fig. 5d, e, f. Only in variant A do individual Jacobi set components remain. The second ROI is marked in brown in the top center of the original data. A larger reddish Jacobi set component can be seen here, with several smaller components adjacent to it, which should be removed. After applying the loop subdivision, the large Jacobi set component is visible and the small ones are no longer recognizable. When smoothing with the binomial filter, there is hardly any effect in Fig. 5b and the large and small Jacobi set components are still present. In contrast, the effect of smoothing by the Gaussian filter is too large and all components are removed, which is visible in Fig. 5c. When using the CA variant D, the result is similar to the binomial filter, and all components are still present. CA variants B and C do better here, where the large Jacobi set component remains and almost all small components disappear. In CA variant A, only the large component remains, as can be seen in Fig. 1 at the bottom left. The third ROI is marked in yellow in the center of the original data. In this ROI, several small Jacobi set components can be recognized in the original data. In the loop subdivision, several large Jacobi set components are separated from each other as can be seen in Fig. 5a. This indicates that the small components are caused by noise or numerical errors and should disappear. With the smoothing filters, the result is similar to before. The binomial filter smoothes too little and the Gaussian filter too much, whereby in this ROI the Gaussian filter does not remove all the small Jacobi set components, but even creates many new ones. Using the CA variants produces very different results here. In variant C, for example, the Jacobi set components are not separated but connected. In variants B and D, the larger components are separated from each other, but many small components remain. In variant A, the large components of the Jacobi set are completely separated and almost all small components disappear so that the results of the loop subdivision are very similar.

Overall, the visual comparison of the cylinder flow shows that the CA variant A is the best variant. This is also shown in the Tab. 1. To

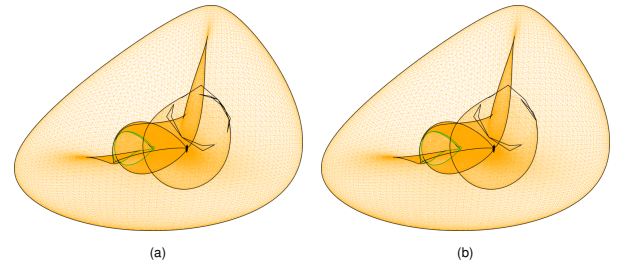


Figure 6: The datasets from Fig. 1 are mapped into the range. The border of the domain is shown in green, the Jacobi sets in black, and the grid in orange for the original data (a) and the dataset after applying CA variant A (b) in orange.

confirm this, the Fig. 1 on the right shows the neighborhood graphs of the Jacobi set components for the original data and for the dataset after applying CA variant A. Here we can see that the algorithm greatly simplifies the Jacobi set components. Besides, the range of the original data is compared with the dataset after applying the CA variant A in Fig. 6 to examine the effects of modifying the data. Despite the differences in the domain, there is hardly any visual difference between the two datasets. The simplification from the Jacobi set components from 679 to 17 using the CA variant A takes 0.2 s.

5.3 Tensile Bar

Next, we look at a collection of real-world datasets, the *Tensile bars* [43, 44, 32]. Various notches or holes are made in these specimens, as shown in Fig. 7, to create different loading conditions and test the material properties. The tensile bars were simulated using the commercial software package Abaqus/Standard CAE [15]. The datasets are unstructured 3D datasets on which 3D symmetric tensor fields of second order are defined. These datasets have a dimension of the volume of the dataset of $120 \times 30 \times 3$. Since the tensile forces are mainly in one plane, we were able to use a single layer from the datasets and triangulate the grid. Here, the 3D tensor is mapped to a 2D tensor from which we derive an invariant set for the comparison. For this comparison, we use the principal invariants I which are the coefficients of the characteristic polynomial. They are defined for 2D tensor fields as $I_1 = \text{tr}T = \lambda_1 + \lambda_2$, and $I_2 = \det T = \lambda_1 \cdot \lambda_2$, where $\text{tr}T$ denotes the trace and $\det T$ the determinant of T for the eigenvalues λ . A detailed description of these and other stress

Table 1: The results for the length of the Jacobi sets, and the number of Jacobi set components can be seen for all datasets and methods tested, with the bolded values being the best. CA is short for Collapse Algorithm.

Dataset / Cells	Method	Length of Jacobi sets	# of Jacobi set components
Cylinder Flow Fig. 1, and Fig. 5 178'702	Original Data	92.4614	679
	Binomial filter	64.9239	448
	Gaussian filter	40.0536	101
	Loop subdivision	82.9961	6'311
	CA Variant A	44.056	17
	CA Variant B	48.8755	41
Tensile Bar A Fig. 8a / 30'960	Original Data	1382.95	817
	Loop subdivision	1845.95	17'858
	CA Variant A	615.601	19
Tensile Bar B Fig. 8b / 22'360	Original Data	1476.09	800
	Loop subdivision	2200.9	23'625
	CA Variant A	767.146	50
Tensile Bar C Fig. 8c / 29'560	Original Data	1503.17	883
	Loop subdivision	2068.83	21'529
	CA Variant A	675.679	18
Tensile Bar D Fig. 8d / 26'850	Original Data	963.471	519
	Loop subdivision	1010.44	7'164
	CA Variant A	465.583	40
Tensile Bar E Fig. 8e / 36'336	Original Data	1004.65	490
	Loop subdivision	1238.53	9'442
	CA Variant A	535.383	38
Tensile Bar F Fig. 8f / 27'210	Original Data	791.891	329
	Loop subdivision	1004.82	7'259
	CA Variant A	478.767	24
Tensile Bar G Fig. 8g / 34'012	Original Data	815.444	326
	Loop subdivision	1094.49	6'827
	CA Variant A	514.535	36
Tensile Bar H Fig. 8h / 28'226	Original Data	782.771	322
	Loop subdivision	846.473	4'484
	CA Variant A	447.887	28
Hurricane Isabel Fig. 9 / 498'002	Original Data	915'064	43'838
	Binomial filter	662'059	27'344
	Gaussian filter	122'871	4'832
	Loop subdivision	801'508	336'823
	CA Variant A	393'343	2'657

tensors can be found in Holzapfel's textbook [22].

In Fig. 8 we visually compare the extracted Jacobi sets in the original data as seen in the upper Fig. 8a–h with the datasets after applying the CA variant A as seen in the Fig. 8i–p. The range area is displayed using color coding, with the saturation indicating how high the stresses in the dataset are. The first thing to recognize is that the Jacobi sets in all datasets are visually simplified after applying the CA variant A and parts with small Jacobi set components could be greatly reduced. Differences can be seen in the datasets with notches and the datasets with holes. For example, Fig. 8a, i shows that the Jacobi sets are simplified at the notches and the structure becomes more recognizable for domain experts. In Fig. 8c, k, it can be seen that in parts where a high density of small Jacobi set components occurs, the Jacobi sets are simplified, but the symmetry of the dataset is partially broken. For example, in Fig. 8e, m, it can be seen that the Jacobi sets are simplified in the part around the hole. In Fig. 8f, n it can also be seen that the Jacobi sets can be simplified

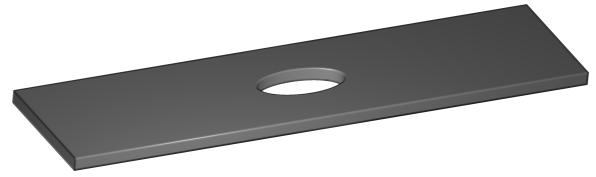


Figure 7: Example of the 3D tensile bar F geometry.

at the upper and lower end of the tie rod where low stress values also appear in the dataset, which can also be seen from the white color. In the Tab. 1 it can be seen that CA variant A can greatly reduce all two measures. The results of the loop subdivision can also be seen here. It is noticeable that the length of all Jacobi sets increases significantly compared to the original data. For the smoothing filters, we have no results for these datasets, as the used filters only work on structured grids. Overall, you can see in the figures that the noise in the datasets can be greatly reduced. The simplification from the Jacobi set components for the tensile bar D from 519 to 40 using the CA variant A takes 0.085 s. All the other tensile bars need less time.

5.4 Hurricane Isabel

Another real-world dataset from the field of meteorology is *Hurricane Isabel*[41, 27], which was published as a freely available dataset as part of the 2004 SciVis contest. This modeled hurricane is based on Hurricane Isabel which caused severe destruction in September 2003. It is a 3D dataset with a resolution of $500 \times 500 \times 100$ and 48 time steps and contains 13 scalar variables, such as velocity, at each data point. We use a layer from the dataset at height 50 at time step 30 and triangulate the grid.

To do this, we look at the scalar fields for pressure and temperature for the corresponding visual analysis in Fig. 9. This figure shows the algorithms to be compared, and the original data. An overall view of the dataset can be seen in Fig. 9c, 9d, 9i, and Fig. 9j. Visually, it can be seen that all 3 algorithms simplify the Jacobi sets compared to the original data. In Fig. 9a, 9e, 9g and Fig. 9k a cutout of the center of the hurricane is shown. It is easy to see that CA variant A and the loop subdivision simplify the Jacobi sets but still retain the structures. In contrast, the Gaussian filter simplifies the center so much that it looks completely different compared to the original data. A visual comparison in Fig. 9j shows that the Gaussian filter greatly simplifies many large structures. A second cutout of the edge of the hurricane can be seen in Fig. 9b, 9f, 9h, and Fig. 9l. Here it can be seen in the original data that many small Jacobi set components occur in the part where the difference between the range areas is small. Visually, these small Jacobi set components have decreased in the loop subdivision as well as in the CA variant A. This also shows that the Gaussian filter greases the data too much. Tab. 1 clearly shows that the CA variant A reduces the components of the Jacobi set the most and that the loop subdivision leads to more Jacobi set components despite an optical simplification. For the Gaussian filter, it can be seen that it reduces the components of the Jacobi sets and even reduces the length of the Jacobi sets the most, but this is at the expense of important structures. Simplifying the Jacobi set components from 43'344 to 2'657 using the CA variant A takes 5.5 s.

6 DISCUSSION

In this study, we compared the collapse algorithm (CA) variants, smoothing filters, and a loop subdivision with the original data using different datasets to investigate the effect on the Jacobi set simplification. Besides, we visually examined the influence of the different neighborhood graphs on the result and compared the best variant with the smoothing filters and the loop subdivision.

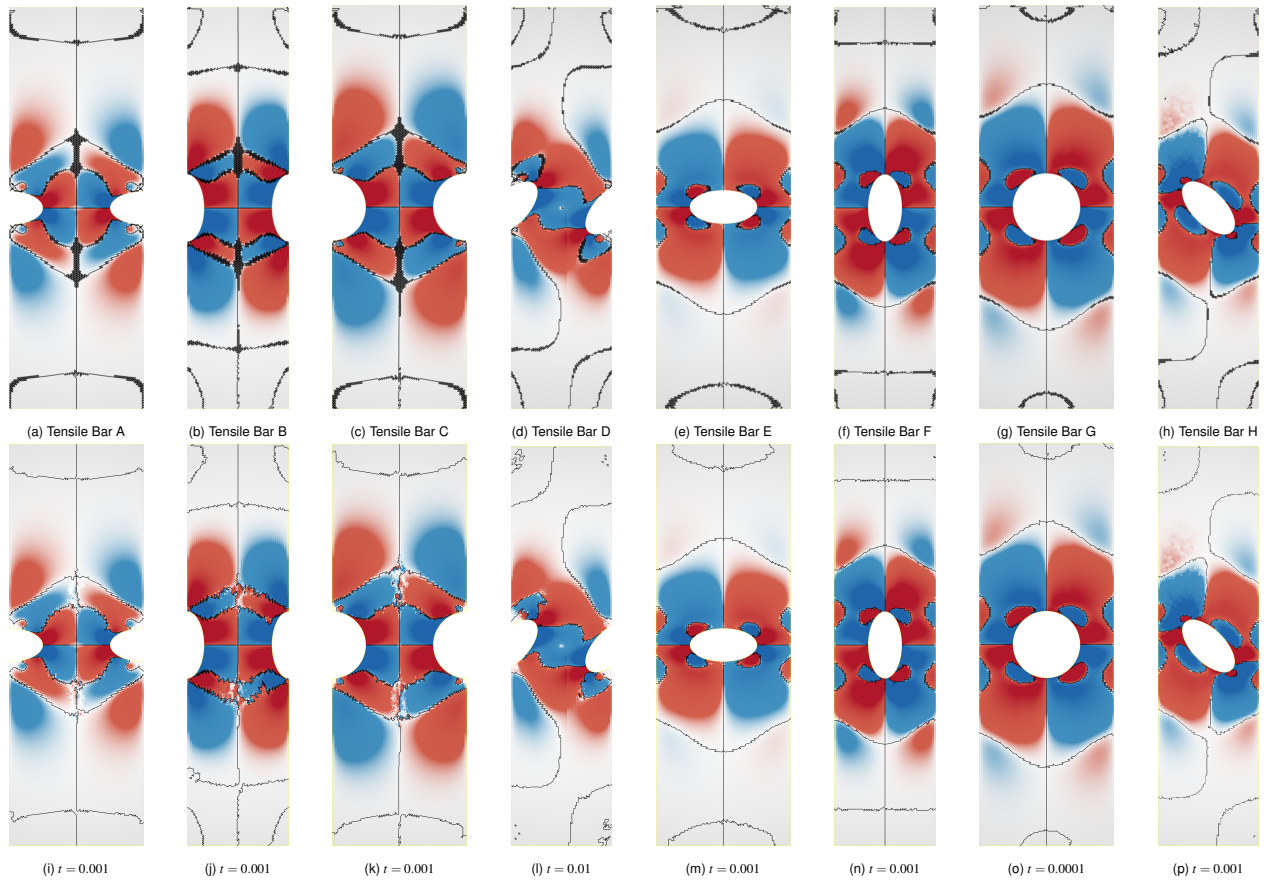


Figure 8: Comparison of the calculated Jacobi sets in the Tensile Bar datasets for the original data in the upper figures and the Collapse Algorithm Variant A with the corresponding τ values shown in the lower figures.

Our investigations have shown that the structure of the neighborhood graph is of great importance, as the results differed from one another. The idea of converting the neighborhood graph into a neighborhood tree had certain advantages, as this made it possible to assign an orientation direction in particularly noisy parts. However, the decision as to whether the negative (variant B) or the positive orientation (variant C) was preferred led to very different results. Therefore, the choice of variant varies depending on the case. The assignment of the cells according to their local neighborhood (variant D) could already solve this problem better but led to the fact that many small Jacobi set components were still not simplified because they were assigned according to their neighborhood. For this reason, the neighborhood graph was used in CA variant A without further adjustments. This has the advantage that the structure is simpler and in the examples considered, the results were not worse, but even better. However, there were small noisy parts in Fig. 1 at the bottom left where not all the noise could be removed. Compared to the original data, the results of the CA variant A are very good. The neighborhood graphs can be greatly simplified, which is particularly beneficial for domain experts, as the analysis becomes more difficult as the data gets larger and larger. This can also be seen when looking at all datasets together, whereby CA variant A was able to reduce the number of Jacobi set components by more than an order of magnitude on average compared to the original data. The length of the Jacobi sets and the number of Jacobi set components were reduced by more than half on average with CA variant A. The visual results also show that the dataset provides a significantly better overview. The comparison of the CA variant A with the loop subdivision showed that the CA does not affect the

essential structures in the dataset and can simplify the datasets at the same time. However, there were still deficits, particularly in the case of data with high symmetry and very noisy parts. Here, the CA should be further optimized to deliver even better results. Although the loop subdivision can visually provide very good results, the actual problem of noisy small Jacobi set components is not solved, but the triangles where this occurs become smaller, as already shown in the work of Klötzel et al. [27] can be seen. The comparison with the Gaussian filter, showed that although it removes common noise in the green regions of interest very well, the essential structures are also changed and thus the content is lost. This can also be seen in Fig. 9k where the Gaussian filter can simplify very noisy parts, leaving many small Jacobi set components. Further comparative tests with different smoothing filters can be found in the appendix.

Overall, it can be seen that the CA variant A can greatly simplify the Jacobi sets without removing essential structures and achieves better results compared to the other algorithms. Since the algorithm collapses cells, this can lead to the fact that a component cannot be completely removed if the algorithm runs into an oscillation, and at this point, the neighborhood graph cannot be simplified further. Examining the runtime in all datasets, it can be seen that for CA variant A it depends on the number of Jacobi set components to be reduced and requires an average of 13.80ms to reduce 100 Jacobi set components.

7 CONCLUSION & FUTURE WORK

In this study, an algorithm for simplifying Jacobi sets for bivariate 2D scalar fields in unstructured triangulated grids was presented by adjusting the underlying data. This algorithm is based on the

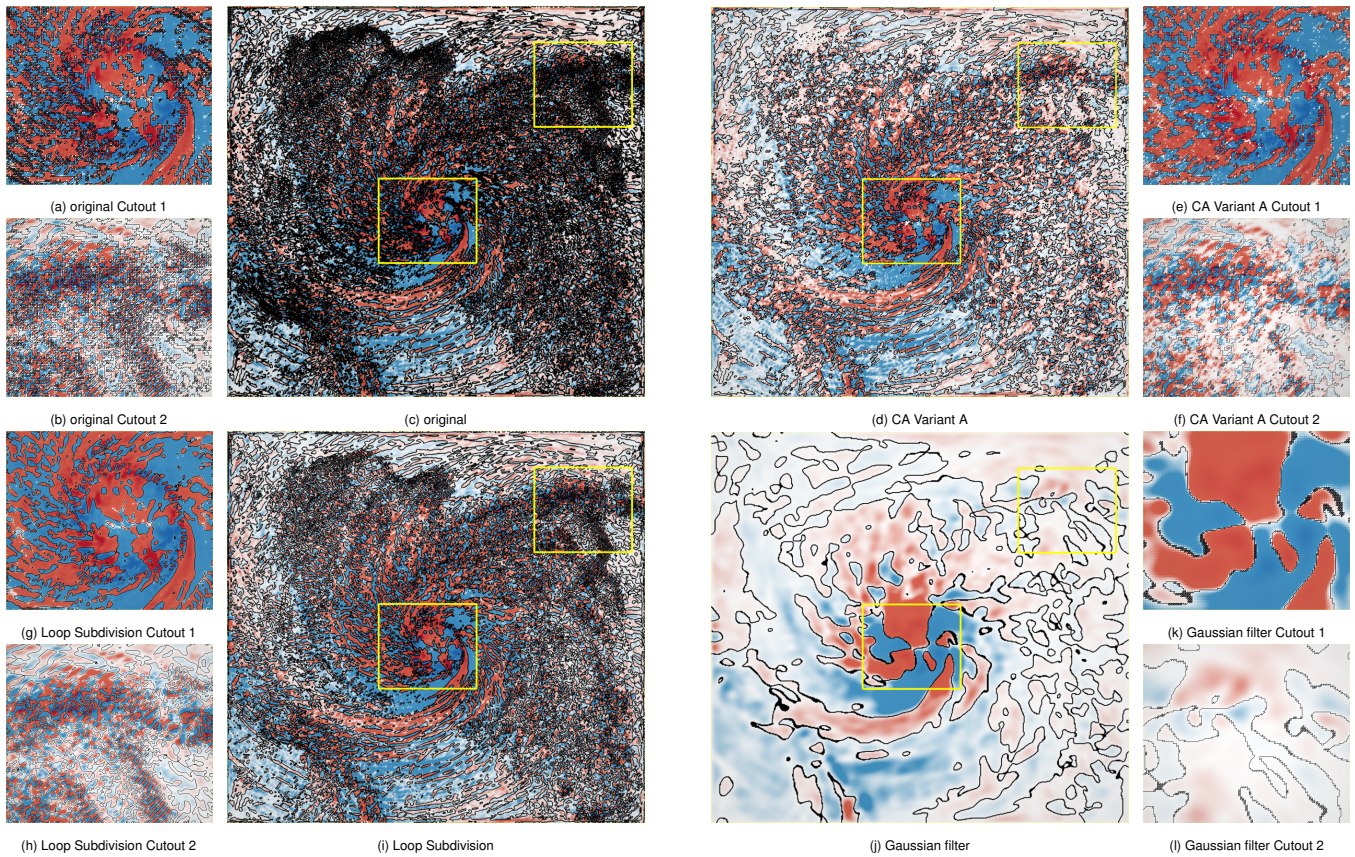


Figure 9: Comparison of the calculated Jacobi sets in the Hurricane Isabel dataset for the original dataset (c), the Collapse Algorithm (CA) Variant A with $t = 200$ (d), the Loop Subdivision (i), and the Gaussian filter (j). Additional comparison of the four methods in detail for cutout 1 (coordinates (772, 683) – (1278, 1122)) (a) (e) (g) (k) and cutout 2 (coordinates (1572, 1382) – (2077, 1822)) (b) (f) (h) (l).

collapsing of cells which changes the underlying data, whereby the collapsing cells are identified using a neighborhood graph. This resulted in four algorithm variants. How well the algorithm variants, smoothing filters, and loop subdivision can simplify the Jacobi sets was then studied with the original data using three datasets from the literature. The best of the four variants was determined, which is the collapse algorithm variant A, especially with regard to the number of components of the Jacobi set, which was an order of magnitude lower than in the original data. The Gaussian filter, on the other hand, was able to reduce the length of the Jacobi sets the most, but changed the dataset too much. However, the collapse algorithm was also able to reduce the length of the Jacobi sets by half. The results of the collapse algorithm are promising, although some adjustments are still needed for symmetric data such as the tension rods to better preserve symmetry. Besides, an adapted Jacobi set visualization was presented to account for degenerate cells and assign them to a Jacobi set component according to their neighborhood.

As indicated at various points in the paper, there is still a lot of potential for future work: Currently, a threshold is used to select Jacobi set components to be collapsed, here an automatic selection that works over the neighborhood could be a possible extension. Furthermore, the presented collapse algorithm has the potential for accelerating the extraction of fiber surfaces or lines, especially if the extraction is done using Jacobi sets, as in the work of Sharma et al. [34]. This further development would not only increase the extraction speed but also simplify the visual analysis of the range. Symmetrical datasets show optimization possibilities for the algorithm at various points, as the assignment of collapsed cells cannot always be unambiguous. One approach would be to adapt the algo-

rithm so that cells do not collapse directly, but collapse gradually. This could achieve a smooth transition between the cells of the regions to be collapsed. An obvious next step is to adapt the algorithm to work with 3D bivariate datasets, as the domain still consists of triangles. However, this requires further investigation and evaluation of different datasets to investigate the influence on the domain. Especially the selection of the Jacobi set components to be collapsed can be difficult. Furthermore, the focus is on extending the algorithm for 3D case. Here, it would first have to be investigated how the collapsing of tetrahedra can be realized and what effects this has on the dataset. This would introduce a new dimension of complexity and require careful investigation and testing with different datasets.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SCHE 663/17-1. The authors will like to thank Markus Stommel of the Leibniz Institute of Polymer Research Dresden, Germany for providing the tensile bar datasets. The authors will like to thank Bill Kuo, Wei Wang, Cindy Bruyere, Tim Scheitlin, and Don Middleton of the U.S. National Center for Atmospheric Research (NCAR), and the U.S. National Science Foundation (NSF) for providing the Weather Research and Forecasting (WRF) Model simulation data of Hurricane Isabel. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research “Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig”, project identification number: SCADS24B.

REFERENCES

- [1] I. V. Artamonova, V. V. Alekseev, and N. G. Makarenko. Gradient measure and jacobi sets for estimation of interrelationship between geophysical multifields. *Journal of Physics: Conference Series*, 798(1):012040, jan 2017. doi: 10.1088/1742-6596/798/1/012040 1
- [2] T. P. Barnett, D. W. Pierce, and R. Schnur. Detection of anthropogenic climate change in the world’s oceans. *Science*, 292(5515):270–274, 2001. doi: 10.1126/science.1058304 1
- [3] 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 2
- [4] P. Bremer, E. Bringa, M. Duchaineau, A. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci. Topological feature extraction and tracking. *Journal of Physics: Conference Series*, 78(1):012007, 2007. doi: 10.1088/1742-6596/78/1/012007 1, 2
- [5] P.-T. Bremer, B. Hamann, H. Edelsbrunner, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004. doi: 10.1109/TVCG.2004.3 2
- [6] G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pp. 184–193, 2007. doi: 10.1007/s00454-009-9176-0 2
- [7] H. Carr and D. Duke. Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1100–1113, 2013. doi: 10.1109/PacificVis.2013.6596141 2
- [8] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003. doi: 10.1016/S0925-7721(02)00093-7 2
- [9] H. Carr, J. Snoeyink, and M. Van De Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Visualization 2004*, pp. 497–504. IEEE, 2004. doi: 10.1109/VISUAL.2004.96 2
- [10] 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. doi: 10.1016/j.comgeo.2006.05.009 2
- [11] CGL, ETH Zurich. CGL @ ETHZ - Data. <https://cgl.ethz.ch/research/visualization/data.php>. Accessed: 2024-01-01. 5
- [12] A. Chattopadhyay, H. Carr, D. Duke, Z. Geng, and O. Saeki. Multivariate topology simplification. *Computational Geometry*, 58:1–24, 2016. doi: 10.1016/j.comgeo.2016.05.006 2
- [13] A. Chattopadhyay, H. A. Carr, D. J. Duke, and Z. Geng. Extracting jacobi structures in reeb spaces. In *EuroVis (Short Papers)*, 2014. doi: 10.2312/eurovisshort.20141156 2
- [14] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pp. 263–271, 2005. doi: 10.1007/s00454-006-1276-5 2
- [15] Dassault Systèmes. Abaqus/Standard. <https://www.3ds.com/products/simulia/abaqus/standard>. Accessed: 2024-03-01. 6
- [16] Edelsbrunner, Harer, and Zomorodian. Hierarchical morse—small complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, 30:87–107, 2003. doi: 10.1007/s00454-003-2926-5 2
- [17] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002. doi: 10.1109/SFCS.2000.892133 2
- [18] H. Edelsbrunner and J. Harer. Jacobi sets of multiple morse functions. *Foundations of computational mathematics, Minneapolis*, 8:35–57, 2002. 1, 2, 3
- [19] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Local and global comparison of continuous functions. In *IEEE Visualization 2004*, pp. 275–280. IEEE, 2004. doi: 10.1109/VISUAL.2004.68 1, 2, 3
- [20] X. He, Y. Tao, Q. Wang, and H. Lin. Multivariate spatial data visualization: a survey. *Journal of visualization*, 22:897–912, 2019. doi: 10.1007/s12650-019-00584-3 2
- [21] 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 2
- [22] A. G. Holzapfel. *Nonlinear solid mechanics II*. John Wiley & Sons, Inc., 2000. 7
- [23] L. Huettenberger, C. Heine, H. Carr, G. Scheuermann, and C. Garth. Towards multifield scalar topology based on pareto optimality. *Computer Graphics Forum*, 32(3pt3):341–350, 2013. doi: 10.1111/cgf.12121 2
- [24] L. Huettenberger, C. Heine, and C. Garth. Decomposition and simplification of multivariate data using pareto sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2684–2693, 2014. doi: 10.1109/TVCG.2014.2346447 2
- [25] F. Iuricich, S. Scaramuccia, C. Landi, and L. De Floriani. A discrete morse-based approach to multivariate data analysis. In *SIGGRAPH ASIA 2016 Symposium on Visualization*, pp. 1–8, 2016. doi: 10.1145/3002151.3002166 1
- [26] C. Jung, T. Tél, and E. Ziemniak. Application of scattering chaos to particle transport in a hydrodynamical flow. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 3(4):555–568, 1993. doi: 10.1063/1.165960 5
- [27] 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, 2022. doi: 10.1007/s00371-022-02557-4 2, 7, 8
- [28] C. Loop. *Smooth subdivision surfaces based on triangles*. PhD thesis, University of Utah, 1987. 5
- [29] C. Luo, I. Safa, and Y. Wang. Approximating gradients for meshes and point clouds via diffusion metric. *Computer Graphics Forum*, 28(5):1497–1508, 2009. doi: 10.1111/j.1467-8659.2009.01526.x 2
- [30] K. Makela, T. Ophelders, M. Quigley, E. Munch, D. Chitwood, and A. Dowtin. Automatic tree ring detection using jacobi sets. *arXiv preprint arXiv:2010.08691*, 2020. doi: 10.48550/arXiv.2010.08691 1
- [31] G. Norgard and P.-T. Bremer. Ridge–valley graphs: Combinatorial ridge detection using jacobi sets. *Computer Aided Geometric Design*, 30(6):597–608, 2013. doi: 10.1016/j.cagd.2012.03.015 1
- [32] F. Raith, B. Nsonga, G. Scheuermann, and C. Heine. Fast fiber line extraction for 2d bivariate scalar fields. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pp. 156–160. IEEE, 2023. doi: 10.1109/VIS54172.2023.00021 6
- [33] G. Reeb. Sur les points singuliers d’une forme de pfaff complètement intégrable ou d’une fonction numérique [on the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946. 2
- [34] 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 1, 9
- [35] G. Singh, F. Mévoli, G. E. Carlsson, et al. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *PBG@ Eurographics*, 2:091–100, 2007. doi: 10.2312/SPBG/SPBG07/091-100 2
- [36] B. Strodthoff and B. Jüttler. Layered reeb graphs for three-dimensional manifolds in boundary representation. *Computers & Graphics*, 46:186–197, 2015. doi: 10.1016/j.cag.2014.09.026 2
- [37] N. Suthambhara and V. Natarajan. Simplification of jacobi sets. In *Topological Methods in Data Analysis and Visualization*. Springer, 2009. doi: 10.1007/978-3-642-15014-2.8 1, 2
- [38] J. Tierny and H. Carr. Jacobi fiber surfaces for bivariate reeb space computation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):960–969, 2016. doi: 10.1109/TVCG.2016.2599017 2
- [39] 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 1
- [40] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM transactions on graphics (TOG)*, 22(3):445–452, 2003. doi: 10.1145/882262.882290 2
- [41] University Corporation for Atmospheric Research. Hurricane Isabel WRF Model Data Visualization. <https://www.earthsystemgrid.org/dataset/isabeldata.html>. Accessed: 2024-02-01. 7
- [42] M. Van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal.

In *Proceedings of the thirteenth annual symposium on Computational geometry*, pp. 212–220, 1997. doi: 10.1145/262839.269238 [2](#)

- [43] V. Zobel and G. Scheuermann. Extremal curves and surfaces in symmetric tensor fields. *The Visual Computer*, 34(10):1427–1442, 2018. doi: 10.1007/s00371-017-1450-1 [6](#)
- [44] V. Zobel, M. Stommel, and G. Scheuermann. Visualizing gradients of stress tensor fields. In *Modeling, Analysis, and Visualization of Anisotropy*, pp. 65–81. Springer, 2017. doi: 10.1007/978-3-319-61358-1_4 [6](#)