EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

2D Piecewise Linear Scalar Fields with Invertible Integral Lines

T.L. Erxleben[1] and M. Motejat[2] and C. Rössl[1] and H. Theisel[1]

[1]University of Magdeburg, Germany
[2]Dornheim Medical Images GmbH, Germany

Abstract

Integral lines of the gradient flow are standard features in continuously differentiable scalar fields that enjoy some useful properties: They cover the domain densely, do not split, merge, or intersect, and are therefore invertible. For widely used discretizations of scalar fields, the corresponding polygonal approximations of integral lines do not enjoy these properties anymore. We analyze conditions for integral lines in 2D piecewise linear (PL) scalar fields to be invertible by identifying and classifying critical edges in the underlying triangulation. We show that under mild conditions, every 2D PL scalar field can be transformed into an arbitrarily close PL field with invertible integral lines. We present an algorithm that computes this transformation and apply it to a number of test data sets.

CCS Concepts

• ***Human-centered computing*** → ***Scientific visualization;*** • ***Computing methodologies*** → *Mesh models;*

1. Introduction

Computing integral lines in scalar fields is a standard problem in visualization that has a variety of applications, like the segmentation of a scalar field by computing the Morse-Smale (MS) complex. Given a 2D continuously differentiable scalar field $s(\mathbf{x})$ in the spatial domain $D \subseteq \mathbb{R}^2$, a parametric curve $\mathbf{p}(t)$ in D is an *integral line* of s if

$$\dot{\mathbf{p}}(t) := \frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \nabla s(\mathbf{p}(t)) \tag{1}$$

for all t, where ∇s is the gradient of s. Regarding s as a height field, integral lines describe the steepest uphill or downhill flow of a mass-less particle under (inverted) gravity.

Integral lines of continuously differentiable scalar fields enjoy a number of useful properties: they cover D densely, i.e., for every $\mathbf{x} \in D$ there exists exactly one integral line passing through $\mathbf{x}$. Integral lines do not merge or cross. Moreover, integral lines are *invertible*: consider an integral line $\mathbf{p}(t)$ starting at $\mathbf{x}$, i.e., $\mathbf{p}(0) = \mathbf{x}$, and let $\mathbf{y} = \mathbf{p}(t_1)$ be an arbitrary point on $\mathbf{p}$ for some t_1. Then $\mathbf{x}$ is on the integral line $\mathbf{q}(t)$ that starts at $\mathbf{y}$, i.e. $\mathbf{q}(0) = \mathbf{y}$, as $\mathbf{q}(-t_1) = \mathbf{x}$. In particular, the integral lines $\mathbf{p}$ and $\mathbf{q}$ are identical curves that differ only in their parametrization. This means that a forward integration over certain time followed by a subsequent backward integration over the same time always ends up at the starting point. Finally, integral lines have a well-defined asymptotic behavior: for $t \to \pm\infty$, they converge to local maxima or minima of s, unless they leave D before. If s is a Morse function, i.e., all of its critical points are non-degenerate, the Morse-Smale complex, i.e., a segmentation of D into areas of similar asymptotic behavior of the integral lines, can be obtained by computing integral lines starting in the neighborhood of saddle points.

Besides these useful properties, integral lines of continuously differentiable scalar fields come with a serious drawback: typically no closed-form solution for non-trivial scalar fields exist, such that integral lines can only be computed by a numerical integration. This adds computational uncertainty [BHJ*14]: the computed integral line does not only depend on the chosen numerical integration method and its parameters. It also requires a heuristic decision about when to stop the integration, since we cannot integrate until infinity. This makes numerically integrated integral lines unreliable when it comes to guarantees about their asymptotic behavior.

A common approach to cope with the computational uncertainty of integral lines is to consider a discretization of both the scalar field and the integral lines. Typical choices are *piecewise linear* (PL) scalar fields defined w.r.t. a regular triangulation of D [EHZ03], or the application of Forman's discrete Morse theory [For01]. For PL fields, there exist two options for computing discrete integral lines: either one traverses trough triangles in the direction of steepest ascent (or descent) with the assumption that the gradient of s is constant within a triangle [BEHP04]. Or one restricts the gradient to the edges of the triangulation and traverses along edges with maximal (or minimal) difference of the scalar values at their vertices [EHZ03]. In both cases, discrete integral lines are polylines that end in local maxima (or minima) of s after a finite number of steps. In the context of discrete Morse theory, integral lines are finite sequences of cells, i.e., sequences of vertices, edges, and faces.

All these discretizations of integral lines remove computational uncertainty. For PL fields and edge-following integral lines as well

as for discrete Morse theory, computation of integral lines uses a purely combinatorial way. For PL fields with integral lines across triangles, integer arithmetic can be applied to do the exact integration, even though the size of the integers tends to increase with every integration step. Also, all discrete integral lines terminate after a finite number of steps.

However, all discretized versions of integral lines mentioned above have a fundamental limitation: they can merge, split, and, in certain cases, cross. Thus, they are not invertible. While these drawbacks are unavoidable because they are inherent to the discrete nature of the methods, intensive research has been conducted to minimize and control their impact [GBP19].

In this paper, we introduce a discrete representation of integral lines that includes the "best of both worlds", i.e., the advantages of both continuous, and discrete integral lines: no numerical uncertainty (if integer arithmetic is employed), finite number of integration steps, no merging, splitting, or crossing, and thus invertible integral lines. Our representation is based on PL fields and integrating piecewise constant gradients in triangles, and we ensure that all integral lines cannot split or merge, and are therefore invertible. Our main contribution is an algorithm that transforms any PL field s into another PL field s' that is arbitrarily close to s but is guaranteed to only have invertible integral lines. The new scalar field s' is computed by applying a finite number of local operations to s.

Table 1 summarizes properties of existing methods in PL scalar fields and our new approach to computing integral lines. Desirable properties are highlighted green, undesirable ones are highlighted red. Also note that in the continuous setting, integral lines are always invertible in theory. In practice however, forward-backward integration may be unstable with arbitrarily large errors causing crossing integral lines.

Figure 1 illustrates the different properties. The MS complex of a PL scalar field as computed by TTK [MBF*21] (figure 1a) shows merging integral lines. Crossing integral lines occur when tracing ILs through triangles and on edges, between forward and backward integrated lines (figures 1b and 1c). Our method resolves those degenerated cases by transforming the field such that all integral lines are invertible (figure 1d). The integral lines can be arbitrarily close but do not merge or cross (see close-ups figures 1e and 1f of the highlighted regions in figure 1d). Also note that, while the example is symmetric along the x-axis (the middle of the function), the topology produced by TTK is not.

	continuous	PL on edges	PL on triangles	discrete Morse	new approach
comp. uncertainty	yes	no	no	no	no
# integration steps	infinite	finite	finite	finite	finite
merging	no	yes	yes	yes	no
crossing	no	yes	yes	no	no
invertible	yes	no	no	no	yes
integration	$\mathbb{R}$	comb.	$\mathbb{Z}$	comb.	$\mathbb{Z}$

Table 1: *Properties of the existing and our new approach to compute integral lines. Integration methods are either numerical schemes ($\mathbb{R}$), combinatorial methods (comb.), or use integer arithmetic ($\mathbb{Z}$).*

2. Related work

Gradient estimation and integral lines Tracing integral lines in piecewise constant scalar fields is a special case of solving differential systems with discontinuous right-hand sides. The numerical integration of these systems is a well-studied problem with a wide range of applications, e.g., in biology [GS02] and control theory [MB97]. We refer to Dieci and Lopez [DL12] for a comprehensive survey of numerical methods. They distinguish two classes of methods: *event-driven methods* and *time-stepping methods*. While time-stepping methods ignore discontinuities and rely on local error estimators, event-driven methods employ a special handling of the (known) discontinuous region. Some methods assume that trajectories *cross* discontinuous regions [Man78], while the extension of the system to a Filippov system [Fil88; PK08] enables solutions that may *slide* on the discontinuous regions. Filippov theory was also applied to discontinuous vector fields to extend vector field topology and identify novel topological features [MDS24].

Computing integral lines by tracing the gradient through the interiors of triangles is a special case to the solution of Filippov systems, in the sense that we define edges that are crossed and identify certain edges, the critical edges, on which integral lines slide.

Mancinelli et al. [MLP19] compare different schemes for gradient estimation on PL scalar fields in 2D to a ground truth and found that piecewise constant gradient fields are sensitive to anisotropic sampling and noise compared to smooth approximations given by gradients estimated at vertices. However, they point out that integral curves in smooth approximations may intersect in practice due to error accumulation in numerical integration. The issue of crossing or merging streamlines of a vector field under numerical integration is addressed in [RS14] which is based on edge maps [BJB*11; BJB*12] that are used to encode the inflow and outflow behavior of a vector field defined over a triangulation. One application of those maps is the extraction of topological features.

Higher order interpolation schemes of gradient fields in combination with edge maps seem like a tempting solution to merging integral lines. However, we identify a major drawback: The relation between the scalar field and its gradient field is lost. It is unclear if the interpolant is even integrable. Note that edge maps are also no solution in piecewise constant vector fields: For an integral line moving on an edge the edge map maps all points to one of its vertices.

Morse-Smale Complexes There are several approaches to computing the Morse-Smale (MS) complex and related Quasi MS complexes. Informally, the MS complex gives a topological segmentation of certain scalar fields, called Morse-Smale functions, into regions of identical gradient flow behavior. The boundaries of those regions, called separatrices, are special integral lines. In the following, we briefly describe PL-based and Forman-based approaches.

Edelsbrunner et al. [EHZ03] present an early boundary-based method for computing Quasi MS complexes of PL scalar fields. In their definition of the Quasi MS complex, the separatrices are no longer required to follow the path of steepest ascent/descent, but only

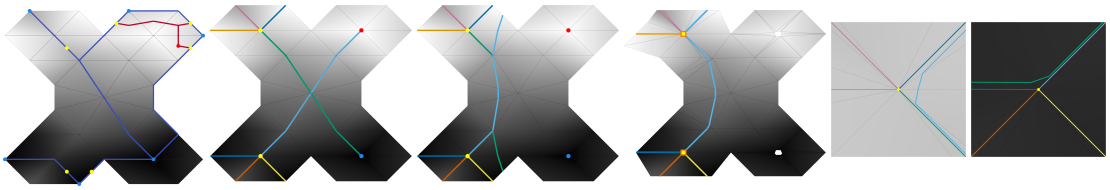

Figure 1: *We demonstrate different discretizations of integral lines and their drawbacks by showing the separatrices of the MS complex of a small PL scalar field. The output of TTK [MBF*21] (a) shows merging intergral lines. Tracing integral lines along edges (b) or through triangles (c) without further restrictions results in crossings. We transform the scalar field using our method resulting in (d). The close-ups (e) and (f) of the highlighted regions in (d) show that integral lines neither merge nor cross.*

to be monotonic, w.r.t. their scalar values. Their rather complicated three-stage algorithm was simplified and improved in [BEHP04] and extended to 3D in [EHNP03]. Bremer and Pascucci [BP07] give a simplified description of an easy to implement algorithm to compute a Quasi MS complex using a boundary-based method. Integration of separatrices is restricted by several rules to avoid degeneracies but merging is still allowed. None of the approaches above resolved the merging of separatrices, instead they simulate an infinitesimal separation distance between them. Those techniques are no general solution as they are restricted to a finite number of lines and do not allow for a proper treatment of all integral lines.

Szymczak and Zhen [SZ12] present a technique to compute the Morse decomposition of a piecewise constant vector field. They extend the vector field using the concept of exploding and imploding edges which are, intuitively, the edges where both incident vectors point towards them or away from them. On those edges integral lines may slide while other edges are crossed. They also applied their algorithm to compute the MS complex of PL scalar fields. In this paper, we define a concept similar to exploding and imploding edges, called critical edges.

Forman [For98; For01] introduced a combinatorial approach to formulate a discretized Morse theory built on simplical complexes. In a *Forman function* each simplex, in contrast to only vertices, is mapped to a scalar value. *Discrete vector fields* are then pairs of vertices and edges, or edges and triangles, such that each simplex is at most part of one pair. Integral curves are discretized as V*-paths*, i.e., a sequence of simplexes of alternating dimensions. When no V-path forms a loop the field is called discrete gradient field and there exists a corresponding Forman function to that field. All methods to compute MS complexes based on this theory construct discrete gradient fields (or analogous Forman functions) from different inputs [ČMD11; GBHP08; MLT*24; GBP19; SN12; GGL*14].

The various Forman-based methods result in different geometric embeddings of the MS complex. Thanh et al. [TAW24] compare Forman-based MS algorithms on different triangulations and give guidelines for well-performing triangulations by including more possible edge directions at each vertex.

Parametrization Invertible integral lines are relevant in several surface and volume parametrization methods and require special treatment. Dong et al. [DBG*06] construct the MS complex of eigenfunctions of the Laplace operator and use its structure to derive a smooth parametrization. Merging separatrices lead to invalid parametrizations and are resolved by iteratively shifting nodes of the complex and topological anti-cancellation of nodes. Myles et al. [MPZ14] work with piecewise linear cross fields and trace invertible integral lines in them to obtain robust mesh parametrizations. Inputs are restricted to fields that are compatible with a quadrangulation, e.g., fields with singularities of index ≥ 1 are forbidden. Invertibility is achieved by using edge maps. Campen et al. [CSZ16] build a piecewise constant vector field over a triangulation/tetrahedrization and use its integral lines to obtain a piecewise linear foliation—a decomposition of the domain in lower dimensional *leaves*— which induces a parametrization. They use the concept of incoming and outgoing simplices during the field construction to ensure invertible integral lines, which is a similar notion to our flow identifier. The constructed field is "simple" by design: integral lines start at the source transversal section and end at the sink transversal section.

While these methods reliably resolve merging separatrices [DBG*06] or even all merging integral lines [MPZ14; CSZ16] they differ in field types and general applicability. This is because, the ultimate goal of these approaches is to compute a parametrization, so changes to the field can be large if needed, e.g., in [MPZ14], which is undesirable if the field itself is subject of analysis. In contrast, our inputs are piecewise linear scalar fields under mild assumptions and the impact of changes to the scalar field can be controlled.

3. Gradients in piecewise linear scalar fields

3.1. Piecewise linear scalar fields

Our method transforms a given piecewise linear (PL) scalar field as input and into another, arbitrarily close, PL scalar field as output. A PL scalar field $s : D \to \mathbb{R}$ is represented by a partition of the domain D into a set of non-degenerate, positively oriented triangles $\mathcal{T}$, which are spanned by vertices $\mathcal{V}$. Each vertex $i \in \mathcal{V}$ is equipped with a position $\mathbf{x}_i \in \mathbb{R}^2$ and a scalar value $s_i \in \mathbb{R}$. The scalar values are interpolated linearly in each triangle. The PL scalar field s is C^0-continuous and has a discontinuous gradient field with a constant gradient vector per triangular piece.

In the following, we refer to triangles as triples $(i,j,k) \in \mathcal{T}$ with $i,j,k \in \mathcal{V}$. The pairs (i,j), (j,k), (k,i) denote the *oriented* edges, also called *half-edges*, which span the triangle (i,j,k). We denote edges (w/o orientation) as 2-sets $\{i,j\}$. Note that each interior edge $\{i,j\}$ of the triangulation is represented by two inversely oriented

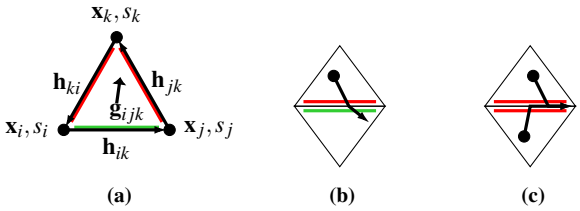

Figure 2: *(a): Triangle (i,j,k) with two outflow half-edges (j,k) and (k,i) and one inflow half-edge (i,j). (b): An integral line crosses a non-critical edge. (c): Two integral lines merge at a critical edge.*

half-edges (i,j) and (j,i). We define the *half-edge vectors* as

$$\mathbf{h}_{ij} = \mathbf{x}_j - \mathbf{x}_i \ .$$

We will consider individual triangles and their neighborhoods, e.g., two triangles sharing an edge or all triangles sharing a vertex i. The latter are spanned by the center vertex i and its 1-ring neighbors $\{j \in \mathcal{V} : (i,j,k) \in \mathcal{T}\}$. For the sake of a concise indexing, we may w.l.o.g. relabel vertices when reviewing local configurations (e.g., enumerate 1-ring neighbors in counter-clockwise order). Also, we write triples (i,j,k) and pairs (i,j) short as ijk and ij, respectively, whenever this is beneficial.

3.2. Gradient inside a triangle

For each linear piece of s supported by a triangle (i,j,k), we compute the constant gradient vector $\mathbf{g}_{ijk} := \nabla s|_{ijk}$ as follows. Let $\left[\mathbf{h}_{ij}\,\mathbf{h}_{jk}\right] \in \mathbb{R}^{2\times 2}$ denote a matrix of concatenated column vectors, then the area of the triangle is

$$A_{ijk} = \frac{1}{2}\det\left[\mathbf{h}_{ij}\,\mathbf{h}_{jk}\right] = \frac{1}{2}\det\left[\mathbf{h}_{jk}\,\mathbf{h}_{ki}\right] = \tfrac{1}{2}\det\left[\mathbf{h}_{ki}\,\mathbf{h}_{ij}\right] > 0 \ .$$

The constant gradient vector is determined as

$$\mathbf{g}_{ijk} := \nabla s|_{ijk} = \frac{1}{2A_{ijk}}\left(s_i\,\mathbf{h}_{jk}^{\perp} + s_j\,\mathbf{h}_{ki}^{\perp} + s_k\,\mathbf{h}_{ij}^{\perp}\right) , \qquad (2)$$

where $\mathbf{h}^{\perp}$ denotes a counter-clockwise rotation by $\frac{\pi}{2}$ of a vector $\mathbf{h}$. The derivation of $\mathbf{g}_{ijk}$ requires the partial derivatives of the linear basis functions and is well-known, see, e.g., [BKP*10].

Preconditions Our method requires two mild assumptions on gradient vectors $\mathbf{g}_{ijk}$ as preconditions:

A1 $\mathbf{g}_{ijk}$ is not parallel to the half-edge vectors $\mathbf{h}_{ij}$, $\mathbf{h}_{jk}$, or $\mathbf{h}_{ki}$, and
A2 $\mathbf{g}_{ijk}$ is not orthogonal to the half-edge vectors $\mathbf{h}_{ij}$, $\mathbf{h}_{jk}$, or $\mathbf{h}_{ki}$.

The second assumption is identical to the common assumption that no two scalar values of an edge are equal, i.e., $s_i \neq s_j$ for any edge $\{i,j\}$ [EHZ03]. Both assumptions are mild in the sense that their violation is structurally unstable: if they are not fulfilled, they can easily be enforced by a small perturbation of s_i, s_j, s_k. In particular, they guarantee that gradients $\mathbf{g}_{ijk}$ do not vanish.

Flow behavior We define the *flow identifier* q_{ij} to describe the inflow/outflow behavior of the gradient across the half-edge (i,j):

$$q_{ij} = -\mathbf{g}_{ijk}^{\top}\mathbf{h}_{ij}^{\perp} \qquad (3)$$

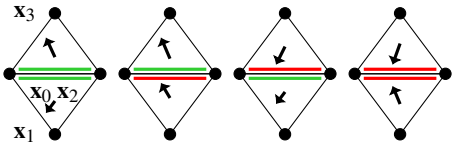

Figure 3: *Four cases of inflow/outflow behavior across the edge $\{0,2\}$ with $s_0 > s_2$.*

and similarly for q_{jk} and q_{ki}. The flow identifier allows for a classification of half-edges: (i,j) is an *outflow* half-edge iff $q_{ij} > 0$, and it an *inflow* half-edge iff $q_{ij} < 0$. (The case $q_{ij} = 0$ is excluded due to assumption A1.) Figure 2a illustrates a triangle (i,j,k) with two outflow half-edges (j,k) and (k,i), and one inflow half-edge (i,j). In this figure and the rest of paper, we color outflow half-edges in red and inflow half-edges in green.

Clearly, inside a triangle (i,j,k), integral lines do not merge or split, and are therefore invertible. We use the flow identifier to distinguish cases on edges and vertices.

3.3. Gradients on edges

PL fields are discontinuous at edges and thus gradients are undefined on edges. We can, however, study the behavior of the gradient flow across the edges. Given are the two triangles $(0,1,2)$ and $(0,2,3)$ of the triangulation that share the common edge $\{0,2\}$ spanned by vertices 0 and 2. Then the gradients of the triangles are not independent but in relation

$$\mathbf{h}_{02}^{\top}\,\mathbf{g}_{012} = \mathbf{h}_{02}^{\top}\,\mathbf{g}_{023}, \qquad (4)$$

i.e., $\mathbf{g}_{012}$ and $\mathbf{g}_{023}$ have the same projection onto the half-edge $\mathbf{h}_{02}$, which follows immediately from equation (2) and orthogonality of $\mathbf{h}_{02}$ and $\mathbf{h}_{02}^{\perp}$. This enables a classification of an edge $\{0,2\}$ by the inflow/outflow behavior of the associated half-edges $(0,2)$ and $(2,0)$ into the four cases in-in, in-out, out-in, out-out that correspond to the conditions $(q_{20} < 0 \wedge q_{02} < 0)$, $(q_{20} < 0 \wedge q_{02} > 0)$, $(q_{20} > 0 \wedge q_{02} < 0)$, $(q_{20} > 0 \wedge q_{02} > 0)$, respectively. Figure 3 illustrates the four cases assuming w.l.o.g. $s_0 > s_2$.

The behavior of integral lines across the edge $\{0,2\}$ with an in-out or out-in classification is not critical, as integral lines cross the edge without merging or splitting. Contrary, the behavior of integral lines across the edge $\{0,2\}$ with an in-in or out-out classification is *critical*, as the integral lines can split and merge at the edge and are therefore not invertible.

Definition 1 An edge $\{i,j\}$ of the triangulation is called *critical edge* if the half-edges (i,j) and (j,i) have the inflow/outflow classification out-out or in-in, i.e., $q_{ij}\,q_{ji} > 0$.

Figure 2b illustrates the crossing of integral lines starting from $\mathbf{x}_0$ over a non-critical edge. Figure 2c illustrates the merging of two integral lines starting from $\mathbf{x}_0$ and $\mathbf{x}_1$ along a critical edge.

3.4. Gradients on vertices

As for edges, the gradient is undefined at vertices, but we can study their local behavior near vertices. We consider the gradient for a

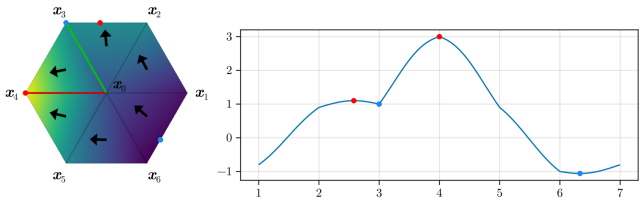

Figure 4: *Vertex 0 of degree 6 and its slope function $a(t)$.*

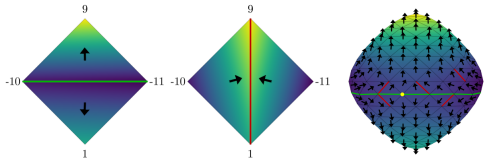

Figure 5: *(a): Two triangles sharing a critical edge. (b): An edge flip does not resolve the critical edge. (c): Repeated Butterfly subdivision does not resolve the critical edge neither.*

local configuration with the center vertex 0 of degree n and its 1-ring neighbors $1,...,n$ enumerated counter-clockwise. We define the slope function $a(t)$ that captures the slope of $s(\mathbf{x})$ in every direction around the center vertex 0 at $\mathbf{x}_0$ with scalar value s_0 by walking the cyclic polygon $(\mathbf{x}_1,\dots,\mathbf{x}_n,\mathbf{x}_{n+1}=\mathbf{x}_1)$ as a periodic function defined on the domain $[1,n+1)$ as

$$a(t) = \frac{(1-u)s_i + u\,s_{i+1} - s_0}{\|(1-u)\mathbf{x}_i + u\,\mathbf{x}_{i+1} - \mathbf{x}_0\|} \tag{5}$$

with $i \le t < i+1$ and $i = 1,\dots,n$ with the local coordinate $u = t - i$ (see figure 4). The global maximum and minimum of $a(t)$ denote the directions of the integral line through $\mathbf{x}_0$. Note that $a(t)$ is a piecewise, C^0-continuous function that can only have a finite number of local extrema at the following locations:

- at a vertex i for $t = i$. We call such local extrema *vertex extrema*.
- between two vertices at $(1-u)\,\mathbf{x}_i + u\,\mathbf{x}_{i+1}$ for $0 < u < 1$. We call such local extrema *inner extrema*.

The condition for an inner extremum on the edge $\{i, i+1\}$ is

$$((1-u)\mathbf{x}_i + u\mathbf{x}_{i+1} - \mathbf{x}_0)) \parallel \mathbf{g}_{0,i,i+1}\ . \tag{6}$$

This is a linear condition in u, meaning that there is at most one inner extremum of a on each edge. We observe the following:

Lemma 1 The slope function $a(t)$ of vertex 0 has a local vertex extremum at the vertex i, i.e., for $t = i$, iff $\{0,i\}$ is a critical edge.

Theorem 1 All integral lines of a PL scalar field $s(\mathbf{x})$ are invertible iff no edge in its PL representation is critical.

See Appendix A for proof of lemma 1 and theorem 1. Theorem 1 is the main theoretical contribution of our approach. In general, PL scalar fields have non-invertible integral lines because they may contain critical edges. In the next section, we provide an algorithm for locally resolving critical edges and thus transforming any PL scalar field to a field without critical edges, i.e., with guaranteed invertible integral curves.

Boundary case The considerations in this section apply, as is, only for an *interior* vertex 0. If the center vertex 0 is a *boundary vertex*, a similar function $a(t)$ can be defined along the triangle fan around the center 0. If the global maximum/minimum is located at a boundary edge, integration is stopped at the center vertex 0.

3.5. Classification of interior vertices

Based on its slope function $a(t)$, every interior vertex can be classified by counting the number of roots of $a(t)$. Conditions A1 and A2 guarantee that $a(t)$ has only a limited, particularly finite, number of roots. We call the regions between two roots positive homogeneous regions if $a(t) > 0$ within the region, or negative homogeneous regions if $a(t) < 0$ within the region. This gives the following classification: a vertex is an *extremum* if $a(t)$ has no roots, a *regular point* if $a(t)$ has two roots, a *saddle* if $a(t)$ has four roots, and a *higher order saddle* if $a(t)$ has more than four roots. Based on this classification, we formulate a third condition that needs to be fulfilled by PL fields to be eligible for our approach:

A3 For every interior vertex, every positive homogeneous region of $a(t)$ has a unique maximum, and every negative homogeneous region of $a(t)$ has a unique minimum.

Since $a(t)$ can only have a finite number of local extrema, condition A3 is also mild in the sense that it can be enforced by small perturbations to the scalar values of the involved vertices.

4. Method

In this section, we present our core contribution: an algorithm that transforms a given 2D PL scalar field into a 2D PL scalar field with invertible integral lines. This problem cannot be solved by elementary local operations like edge flips or repeated subdivision. To see this, consider the example depicted in figure 5: two adjacent triangles $(0,1,2)$ and $(0,2,3)$ with $\mathbf{x}_0 = (-1,0)$, $\mathbf{x}_1 = (0,-1)$, $\mathbf{x}_2 = (1,0)$, $\mathbf{x}_3 = (0,1)$, and $s_0 = -10$, $s_1 = 1$, $s_2 = -11$, $s_3 = 9$ share a critical edge $\{0,2\}$. The center image shows that an edge flip creates a new critical edge. Figure 5 (right) shows that a repeated subdivision (here an interpolating Butterfly scheme [DLG90]) creates new critical edges.

Our approach consists of three local operations which handle local extrema, critical edges touching the boundary of the triangulation, and interior critical edges.

4.1. Local extrema

Around local maxima or local minima critical edges are unavoidable because in these regions many integral lines merge to reach a common critical point as their destination. Because of this, we have to cut out a local extremum from the triangulation and replace it by a hole surrounded by a closed polygon with inflow and outflow at the cut boundaries. We explain this w.l.o.g. for a local maximum at $\mathbf{x}_0$ that is surrounded by a 1-ring of vertices located at $\mathbf{x}_1,...,\mathbf{x}_n$ with $s_0 > s_1,...,s_n$, which is illustrated in figure 6. Let $i+1$ denote the next vertex in the cyclic sequence. We replace the triangle

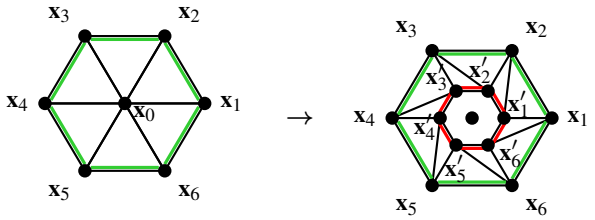

Figure 6: *Replacing a local maximum* $\mathbf{x}_0$ *by a closed outflow polygon* $(\mathbf{x}'_1,\ldots,\mathbf{x}'_n,\mathbf{x}'_1)$.

$(0,i,i+1)$ by two new triangles $(i',i,i+1)$ and $(i',i+1,(i+1)')$ with new vertices i' with positions and scalar values

$$\mathbf{x}'_i = (1-p_i)\,\mathbf{x}_0 + p_0\,\mathbf{x}_i \quad \text{and} \quad s'_i = (1-p_i)\,s_0 + p_0\,s_i \tag{7}$$

for $i = 1,\ldots,n$ and $0 < p_i < 1$. The parameters p_i are chosen such that $\|\mathbf{x}'_1 - \mathbf{x}_0\| = \|\mathbf{x}'_2 - \mathbf{x}_0\| = \cdots = \|\mathbf{x}'_n - \mathbf{x}_0\|$, i.e., all triangles $(0,i,i+1)$ (not shown in the figure) are isoscele triangles. We can then formulate the following:

Lemma 2 Given is an isosceles triangle (i,j,k) with $A_{ijk} > 0$, $\|\mathbf{h}_{ij}\| = \|\mathbf{h}_{ki}\|$ and $s_i > s_j, s_k$. Then (j,k) is an inflow half-edge.

See Appendix A for proof. Lemma 2 directly implies:

Theorem 2 The half-edges $(i+1,i)$ with $i = 1,\ldots,n$ of the hole created by cutting a maximum as above are outflow half-edges.

Therefore, cutting minima/maxima does not change the topology. Figure 6 illustrates cutting a local maximum $\mathbf{x}_0$ of degree 6.

Remark: We may fill the cut-out holes by connecting integral lines from the closed polygon $(\mathbf{x}'_1,\mathbf{x}'_2,\ldots,\mathbf{x}'_n,\mathbf{x}'_1)$ to $\mathbf{x}_0$. While this fully covers the domain with integral lines, we prefer to leave the holes open to show the impact of local extrema.

4.2. Core idea: vertex split

The core idea to resolve a critical edge is to split one of its vertices. Consider the setup of two triangles $(0,1,2)$ and $(0,2,3)$ sharing a critical edge $\{0,2\}$ of type out-out with $s_0 > s_2$ (see figure 7a), as already considered in section 3.3. We split vertex 0 into two new vertices with positions $\mathbf{x}_0^+$ and $\mathbf{x}_0^-$ and assign scalar values s_0^+ and s_0^- (see figure 7b) the following way: We define a unit-length vector $\Delta\mathbf{x}_0$ that satisfies

$$\det[\mathbf{h}_{02}\Delta\mathbf{x}_0] > 0\,, \tag{8}$$

i.e., the positive angle between $\mathbf{h}_{02}$ and $\Delta\mathbf{x}_0$ is smaller than π. Then the new vertex positions and scalar values are defined by

$$\mathbf{x}_0^+ = \mathbf{x}_0 + r\Delta\mathbf{x}_0 \;, \mathbf{x}_0^- = \mathbf{x}_0 - r\Delta\mathbf{x}_0 \tag{9}$$

$$s_0^+ = s_0 + r\Delta s_0 \;, s_0^- = s_0 - r\Delta s_0 \tag{10}$$

with

$$\Delta s_0 = (s_2 - s_0)\,\frac{\Delta\mathbf{x}_0^T\,\mathbf{h}_{02}}{\|\mathbf{h}_{02}\|^2} \tag{11}$$

and some positive number r. This vertex split creates a new triangle $(0^-,2,0^+)$ and transforms the existing triangles $(0,1,2)$, $(0,2,3)$ to the new triangles $(0^-,1,2)$ and $(0^+,2,3)$, respectively.

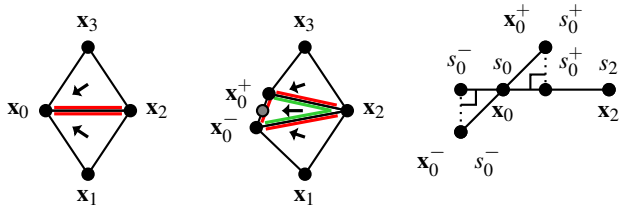

Figure 7: *The core idea to resolve single critical edges is splitting a vertex. In the simple case of two triangles sharing an edge (a), this creates a new triangle and modifies the two existing ones (b). The modified scalars* s_0^-, s_0^+ *are determined by an orthogonal projection of* $\mathbf{x}_0^-, \mathbf{x}_0^+$ *onto the line through* $\mathbf{x}_0$ *and* $\mathbf{x}_2$ *(c).*

The choice of Δs_0 in equation (11) has a geometric interpretation (see figure 7c): s_0^+ is the linear interpolation between s_0 and s_2 at the orthogonal projection of $\mathbf{x}_0^+$ onto the line $(\mathbf{x}_0,\mathbf{x}_2)$, and similar for s_0^-. The vertex split guarantees

$$\mathbf{g}_{0^-,2,0^+} = \frac{s_2 - s_0}{\|\mathbf{h}_{02}\|^2}\,\mathbf{h}_{02}\,, \tag{12}$$

i.e., the gradient $\mathbf{g}_{0^-,2,0^+}$ of the new triangle $(0^-,2,0^+)$ is parallel to the old half-edge $\mathbf{h}_{02}$. Note that equation (12) holds for any positive r. In fact, the gradient of s inside the new triangle $(0^-,2,0^+)$ is independent of r. Equation (12) also gives that, independent of the choice of $r > 0$, the half-edges $(0^-,2)$, $(2,0^+)$ are inflow half-edges, and $(0^+,0^-)$ is an outflow half-edge.

The parameter r controls the area of the new triangle $(0^-,2,0^+)$. While $r > 0$ is required to guarantee $A_{0^-,2,0^+} > 0$, r also has an upper bound: it must be chosen small enough to guarantee that the two transformed triangles $(0^-,1,2)$, $(0^+,2,3)$ have both positive area and the same inflow/outflow behavior as their original counterparts $(0,1,2)$, $(0,2,3)$. If r is small enough to fulfill this as well, we have resolved the critical edge $(0,2)$.

4.3. Critical edges touching the boundary

In the case where the critical edge touches the boundary of the domain of the PL field, the setting is an open triangle fan of the vertices $1,\ldots,n$ around the center vertex 0, and the edges $\{1,0\}$ and $\{0,n\}$ are part of the domain boundary. Suppose the edge $\{0,k\}$ with $1 < k < n$ is a critical edge of type out-out with $s_0 > s_k$. Figure 8 illustrates the setup for $n = 6$ and $k = 4$. Then the vertex split $\mathbf{x}_0 \to \mathbf{x}_0^+, \mathbf{x}_0^-$ as described in equations (9) to (11) creates the new triangle $(0^-,k,0^+)$ and transforms the existing triangles $(0,i,i+1)$ to $(0^-,i,i+1)$ for $i = 1,\ldots,k-1$, and to $(0^+,i,i+1)$ for $i = k,\ldots,n-1$.

To apply the vertex split equations (9) to (11), we have to choose $\Delta\mathbf{x}_0$ and r. We set $\Delta\mathbf{x}_0$ to follow the boundary of the domain of s as close as possible by $\Delta\mathbf{x}_0 = \frac{\mathbf{h}_{i,0}}{\|\mathbf{h}_{i,0}\|}$ where $i = n$ if the angle between $\mathbf{h}_{0,n}$ and $\mathbf{h}_{0,k}$ is smaller than the angle between $\mathbf{h}_{0,1}$ and $\mathbf{h}_{n,0}$. Otherwise we choose $i = 1$. If this does not fulfill equation (8),

$\Delta\mathbf{x}_0$ is reversed. Geometrically, this extends, or shrinks, the length of one of the incident boundary edges.

To compute the upper bound of the parameter r, we have to enforce three groups of conditions. Firstly, positive area of the transformed triangles, i.e., no triangle flip occurs due to the vertex split. Secondly, the same inflow/outflow behavior as their non-transformed counterparts. This is necessary to ensure that no new critical edge is created elsewhere by resolving a critical edge. Thirdly, for critical edges and for minimal/maximal edges starting from vertex 0, no change of the projected gradient flow onto the edge is allowed. Therefore, neither the new vertices $0^-, 0^+$ nor a vertex in their respective 1-ring becomes a new local extremum due to the changes in connectivity combined with increasing/decreasing scalar values at the new vertices.

For preventing triangle flips, we consider the areas of the transformed triangles $A_{0^-,i,i+1}$ for $i = 1,\dots,k-1$ and $A_{0^+,i,i+1}$ for $i = k,\dots,n-1$. For preventing new critical edges, we consider the flow identifiers of the half-edges of all transformed triangles as $q_{0^-,i}, q_{i,i+1}, q_{i+1,0^-}$ for $i = 1,\dots,k-1$ and $q_{0^+,i}, q_{i,i+1}, q_{i+1,0^+}$ for $i = k,\dots,n-1$. For preventing new extrema, we consider the following differences of scalars along the modified edges as

$$\Delta s_i = \begin{cases} s_0^- - s_i & \text{if } q \le i \le k \text{ and } \{0,i\} \text{ is critical, maximal/minimal} \\ s_0^+ - s_i & \text{if } k < i \le n \text{ and } \{0,i\} \text{ is critical, maximal/minimal} \\ 1 & \text{otherwise,} \end{cases}$$

where an edge $\{0,i\}$ is maximal if $s_i \ge s_j$ for $j = 1,\dots,n$, and an edge $\{0,i\}$ is minimal if $s_i \le s_j$ for $j = 1,\dots,n$. Note that all areas $A_{0^-,i,i+1}$, flow identifiers $q_{0^-,i}, q_{i,i+1}, q_{i+1,0^-}$, and scalar differences Δs_i are functions of the parameter r. For $r = 0$, all conditions on area and inflow/outflow behavior are fulfilled, but there is effectively no split as the vertex positions coincide. We want to maximize r, i.e., we search for the smallest positive r, for which one of the desired properties is not fulfilled anymore. Let $r_{A,i}$ be the smallest positive r such that $A_{0^-,i,i+1} = 0$ for $i = 1,\dots,k-1$, or $A_{0^+,i,i+1} = 0$ for $i = k,\dots,n-1$. Let $r_{0,i}$ be the smallest positive r such that either $q_{i,0^-}$ or $q_{i,0^+}$ vanishes. Let $r_{i+1,0}$ be the smallest positive r such that $q_{i+1,0^-}$ or $q_{i+1,0^+}$ vanishes. Let $r_{i,i+1}$ be the smallest positive r such that $q_{i,i+1}$ vanishes. Finally, let r_{s_i} be the smallest positive r such that Δs_i vanishes. Note that finding of the zeros of $A_{0^-,i,i+1}, A_{0^+,i,i+1}, q_{i,i+1}, r_{s_i}$ requires solving a linear equation in r. There always exists one solution, which however, may be negative, i.e., there is no smallest positive r making the expression vanish. In this case, we set r_{Ai} or $r_{i,i+1}$ to $+\infty$. Finding the zeros of $q_{0^-,i}$, $q_{i+1,0^-}$ or $q_{0^+,i}$, $q_{i+1,0^+}$ requires solving a quadratic equation in r. Again, if no smallest positive solution exists, we set $r_{0,i}$ or $r_{i+1,0}$ to $+\infty$. To ensure a finite upper bound and to keep changes to the data small we include r_l which is half of the average edge length of the input triangulation.

With this, the upper bound of r in equations (9) to (11) is

$$\bar{r} = \min \bigcup_{i=1}^{n-1} \{r_{Ai}, r_{0,i}, r_{i+1,0}, r_{i,i+1}, r_{s_i}, r_l\}. \tag{13}$$

With this we know that any r with $0 < r < \bar{r}$ resolves the critical edge $\{0,k\}$ without introducing new critical edges. For all examples

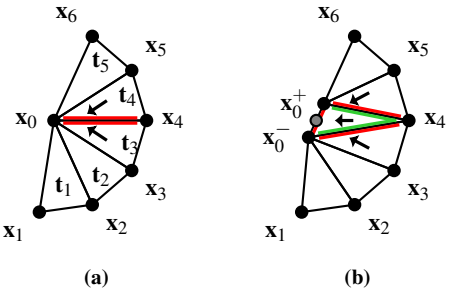

Figure 8: *In open rings, a new triangle is created by the edge split, and the existing $(n-1)$ triangles are modified. In this case, only the width of the splits needs to be chosen, as α is set to best preserve the domain boundary.*

in this paper, we choose the mean

$$r = \tfrac{1}{2}(0 + \bar{r}) \tag{14}$$

as a trade-off between minimizing changes to the scalar function and producing badly-shaped triangles.

4.4. Inner critical edges

The setting for resolving an interior critical edge is a 1-ring of n triangles around a center vertex 0 that is built by the vertices $1\dots,n,1$, similar to section 3.4. We assume that there is no local extremum at $\mathbf{x}_0$, and w.l.o.g. the edge $\{0,1\}$ is a critical edge of type out-out, and $s1 < s0$ (see figure 9a). We consider the scalar function $s(\mathbf{x})$ along the closed polygon $(\mathbf{v}_1,\dots,\mathbf{v}_n,\mathbf{v}_1)$, which is C^0 continuous and piecewise linear. Starting at $\mathbf{x}_1$, we walk the polygon in counter-clockwise direction until we find the first point $\mathbf{x}_p^+$ on the polygon with $s(\mathbf{x}_p^+) = s_0$. Such a point must exist due to the mean value theorem: since $\mathbf{x}_0$ is not an extremum, $s(\mathbf{x})$ along the closed polygon $(\mathbf{x}_1,\dots,\mathbf{x}_n,\mathbf{x}_1)$ contains values above and below s_0. In a similar way we compute $\mathbf{x}_p^-$ by walking on the polygon from $\mathbf{x}_1$ in clockwise direction until we find the first point $\mathbf{x}_p^-$ on the polygon with $s(\mathbf{x}_p^-) = s_0$. Note that $\mathbf{x}_p^-$ and $\mathbf{x}_p^+$ are points on the isoline through vertex 0 and that they must be different points in different triangles. We then select the point $\mathbf{x}_p$ from $\{\mathbf{x}_p^+, \mathbf{x}_p^-\}$ by comparing angles and determine

$$\mathbf{x}_p = \begin{cases} \mathbf{x}_p^- & \text{if } \angle\big((\mathbf{x}_p^- - \mathbf{x}_0), \mathbf{h}_{01}\big) < \angle\big((\mathbf{x}_p^+ - \mathbf{x}_0), \mathbf{h}_{01}\big) \\ \mathbf{x}_p^+ & \text{otherwise} \end{cases}. \tag{15}$$

This choice can be interpreted as follows: $\mathbf{x}_p$ is chosen such that the angle between $\mathbf{h}_{01}$ and $\mathbf{h}_{0p}$ is as small as possible to enforce $\angle\big((\mathbf{x}_p^- - \mathbf{x}_0)\big) < \pi$.

Assume that $\mathbf{x}_p$ lies on the polygon segment spanned by the vertices $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. We split the center vertex 0 according to equations (9) to (11) (see figure 9b). This creates two new triangles

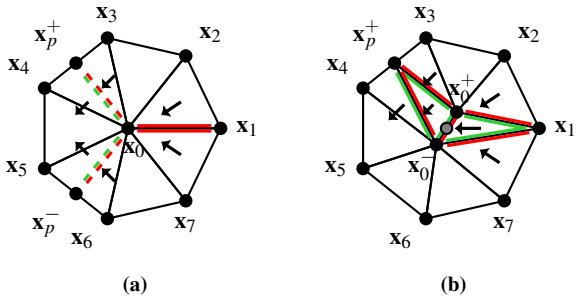

Figure 9: *The vertex split in closed rings creates two new triangles and modifies existing ones. In this setup, we need to choose the split direction and split width.*

$(0^-,1,0^+)$ and $(0^+,p,0^-)$ and transforms the remaining triangles:

$$\begin{aligned}(0,i,i+1) &\to (0^+,i,i+1) \quad \text{for} \quad i=1,\dots,k-1\\ (0,k,p) &\to (0^+,k,p) \quad \text{and} \quad (0,p,k+1) \to (0^-,p,k+1)\\ (0,i,i+1) &\to (0^-,i,i+1) \quad \text{for} \quad i=k+1,\dots,n. \end{aligned} \tag{16}$$

For this vertex split, we chose $\Delta\mathbf{x}_0$ as the angular bisector in positive direction between $\mathbf{h}_{01}$ and $\mathbf{h}_{0p}$. This guarantees that $(0^-,1)$, $(1,0^+)$, $(0^-,0^+)$ are inflow half-edges, and $(0^+,0^-)$ is an outflow half-edge, i.e., the edge $(0^-,0^+)$ is not critical.

Furthermore, we have to find the upper bound of r to guarantee that the transformations equation (16) do not change the orientation of triangles, the inflow/outflow behavior of half-edges, and does not create new extrema. Similar to section 4.3, we consider the areas of all transformed triangles, the flow identifier of all involved half-edges, and the difference of scalars along edges as function of r, and search for the upper bound $\bar{r}$ as the smallest positive r that makes one of these values become zero. This involves finding the roots of linear and quadratic functions. Finally, we set r as in equation (14). With this, we have a local operation that resolves the inner critical edge $(0,1)$ that guarantees to not introduce new critical edges.

4.5. Overall algorithm

Once we have introduced the necessary local operations, we can formulate the algorithm to construct a PL field without critical edges from any PL field that fulfills the conditions A1, A2, A3:

- *Optionally*, apply a topological simplification or smoothing.
- *Optionally*, remove critical edges by edge flipping if possible.
- Identify and cut out local extrema as in section 4.1.
- Identify and resolve critical edges as in sections 4.3 and 4.4.

Since our local operations in sections 4.3 and 4.4 remove a critical edge without creating any new critical edge, our algorithm is guaranteed to terminate after a finite number of iterations, and the result is a PL field without critical edges.

The first two steps are optional: Since critical edges may be due to noise in the data, a simple noise reduction may remove many of or even all of them already. For any critical edge, we can check whether a simple edge flip is able to resolve the critical edge without creating new ones. This may reduce the number of critical edges before the actual algorithm starts. Note, however, that an edge flip may change the linear interpolation significantly. The remaining steps constitute the core algorithm. We remark that the output depends on the order of processing the critical edges. However, the order of processing does not affect the theoretical guarantees, i.e., there always exists a valid solution. We chose a random processing order to be independent of arbitrary edge handles.

Integer Arithmetic Our algorithm relies on integer arithmetic in a sense that all computations can be carried out with ratios of integers, i.e., in the rational numbers $\mathbb{Q}$. So far, we described some steps that need to solve quadratic equations, i.e., the evaluation of the square root. This requires further explanation: Our choices of $\Delta\mathbf{x}_0$ (sections 4.3 and 4.4) involve the normalization to a vector of length 1. While the length of $\Delta\mathbf{x}_0$ is generally not a real number, it only affects the scaling and computation of the upper bound $\bar{r}$. We tighten this upper bound by using a vector $\Delta\mathbf{x}_0$ that has length $1-\varepsilon$ for a small $\varepsilon > 0$. This can be achieved by approximating the square root in the length by an upper bound obtained, e.g., by Heron's algorithm. Similarly, we further tighten the upper bound by using the same approximation when solving the quadratic constraints. A comparable approximation of unit vectors was used by Campen et al. [CSZ16]. Our only requirement on $\Delta\mathbf{x}_0$ is that its angle with $\mathbf{h}_{01}$ is smaller than $\frac{\pi}{2}$. Any direction between $\mathbf{h}_{01}$ and $\mathbf{x}_p$ has this property because $\angle(\mathbf{x}_p-\mathbf{x}_0,\mathbf{h}_{01}) < \pi$. Choosing $\Delta\mathbf{x}_0$ as the angular bisector between $\mathbf{h}_{01}$ and $\mathbf{x}_p-\mathbf{x}_0$ is a natural choice in the continuous setup. Nevertheless, we are allowed to approximate the angular bisector (in local coordinates w.r.t $\mathbf{h}_{01}$) as $\|\mathbf{h}_{01}\|\mathbf{x}_p+\|\mathbf{x}_p\|\mathbf{h}_{01}$ with approximated norms as described above. Alternatively, the rational approximation of polar coordinates from [CSZ16] may be used to approximate the angular bisector. For an inner critical edge (section 4.4) we select one of $\mathbf{x}_p^+$ or $\mathbf{x}_p^-$ based on their angle enclosed with $\mathbf{h}_{01}$. As both angles are within $[0,\pi]$ we can safely compare squared sines, which can be expressed in rational numbers. Thus we compare two rational numbers of the form $\frac{\det^2[\mathbf{a}\,\mathbf{b}]}{\|\mathbf{a}\|^2\|\mathbf{b}\|^2}$ with $\mathbf{a},\mathbf{b}\in\mathbb{Q}^2$.

5. Evaluation

We implemented a prototype of our algorithm in Julia [BEKS17] and used Makie [DK21] for visualization, and a custom half-edge data structure to represent and manipulate triangulations. In theory, all computations can be carried out exactly using rational numbers and integer arithmetic. In practice, this requires the use of an arbitrary precision integer type, though, which comes with a significant performance penalty and is thus infeasible for many practical applications. For this reason, our implementation uses 64-bit precision floating-point arithmetic. Our algorithm repeatedly splits triangles, which can lead to near degenerate triangles. While this is not a problem when using arbitrary precision arithmetic, degenerate triangles become possible if only finite precision arithmetic is available. Figure 14 shows an example, see also section 6.

Experiments For the evaluation of our method, we sampled different test functions for a prescribed triangulation.

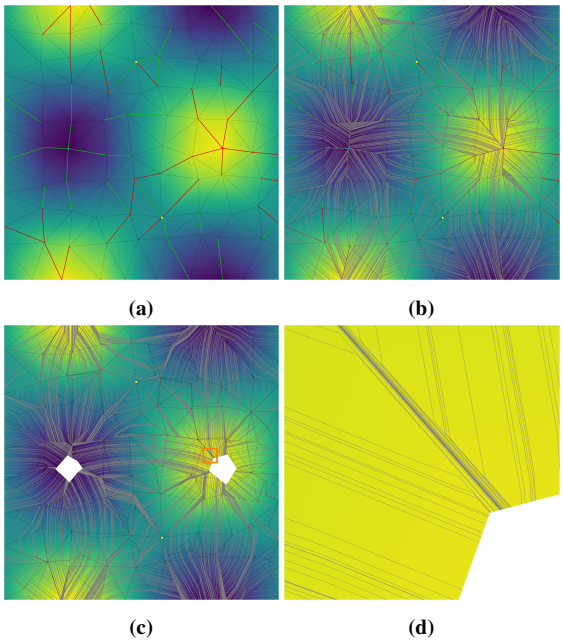

Figure 10: *E1: We apply our method to input (a) generating output (c) without the optional edge-flip step. While original integral lines merge (b), integral lines of our output (c) do not merge and have distinct end points on the boundary as shown in the close-up (d) of the highlighted region in (c).*

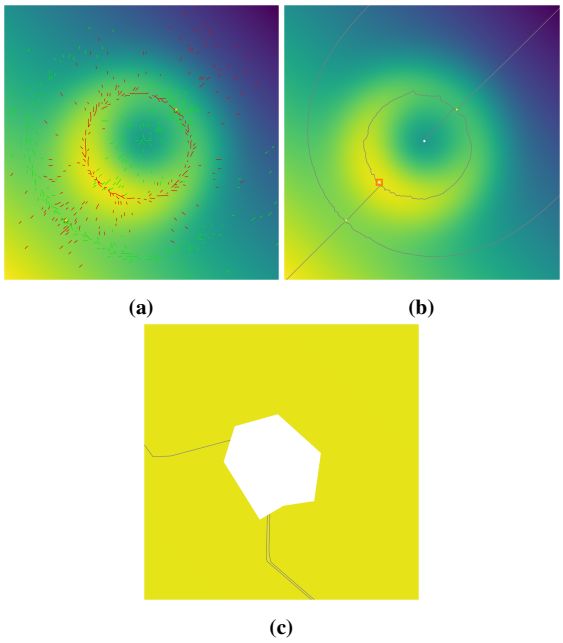

Figure 11: *E2: After applying our method to input (a) we compute the PL MS complex of output (b). The highlighted region in (b) is shown in a close-up in (c) where two distinct endpoints of the separatrices can be seen. This result can also be compared to [TAW24].*

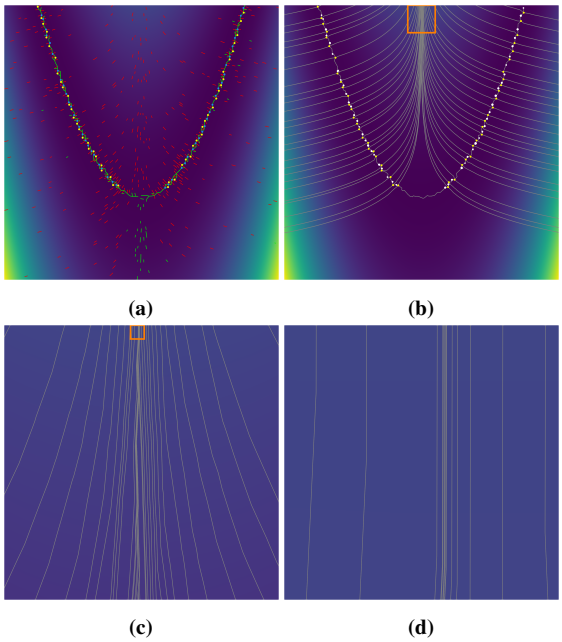

Figure 12: *E3: We apply our method to input (a) resulting in (b) and show its PL MS complex. (c) and (d) show close-ups of the highlighted regions in (b) and (c), respectively. While the distance between the separatrices becomes arbitrary small they do not merge.*

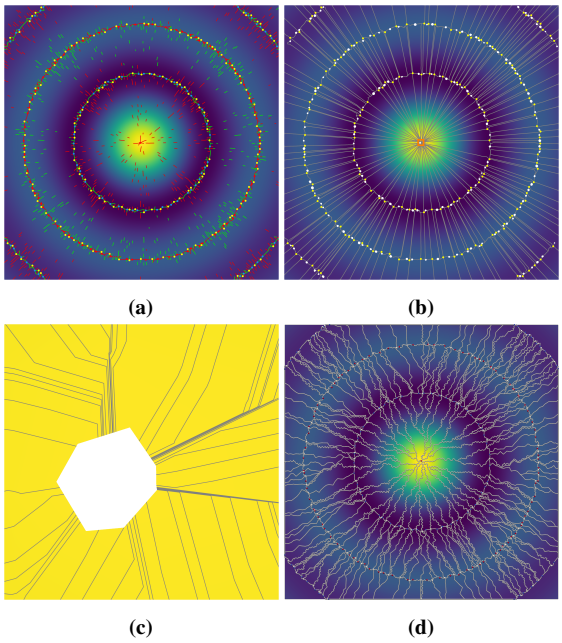

Figure 13: *E4: We apply our method to input (a) resulting in output field (b) and compute its MS complex. Close-up (c) shows the central maximum highlighted in (b). Separatrices have distinct end points on the boundary. The Forman-based MS complex as implemented in the Topology Toolkit shows merging separatrices (d).*

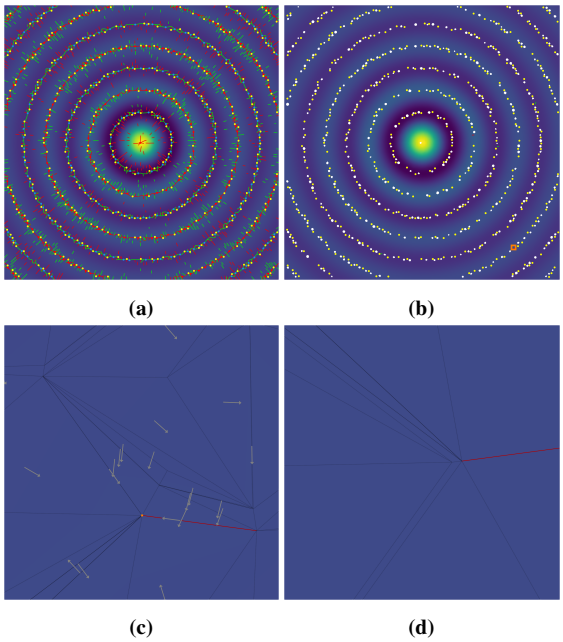

Figure 14: *E5: We sample the same function as in E4 but in a larger region by using a different affine transformation (see table 2). This results in input (a) with more ridges sampled on a lower spatial resolution. Application of our method results in (b). Close-up (c) and (d) show the region highlighted in (b) and (c) respectively. The side-length of (d) is $3 \cdot 10^{-5}$. Gray arrows depict the normalized gradient per triangle. While most triangles are only close to collapsing–i.e., nearly degenerate–some are degenerate w.r.t. to the given numerical precision, and a few critical edges could not be resolved.*

$$f_1(x,y) = \sin(x)\cos(y) \tag{17}$$

$$f_2(x,y) = \exp\left(-2\left(\sqrt{x^2+y^2}-1\right)^2\right) - 0.3\,(x+y) \tag{18}$$

$$f_3(x,y) = (1-x)^2 + 100(y-x^2)^2 \tag{19}$$

$$f_4(x,y) = \operatorname{sinc}\left(\frac{1}{\pi}\sqrt{x^2+y^2}\right) \tag{20}$$

The test function f_1 equation (17) presents a simple introductory example. Example f_2 equation (18) was taken from Reininghaus et al. [RGH*12] and was also used by Thanh et al. [TAW24] in their comparison of different algorithms to compute a MS complex. f_3 in equation (19) is the Rosenbrock function [Ros60], which is a common benchmark and a good test for our method due to its converging separatrices. f_4 in equation (20) posed a difficult input for our method due to its many spurious critical points that are discretization artifacts.

We use a square domain $[-1,1]^2$. All input triangulations are Delaunay triangulations [Van24] of sets of sample points: We sampled the domain using a regular grid of $n \times n$ points with random noise added, where boundary points were restricted to movement on the boundary and the four corner points were fixed.

We used $n = 10$ (D1) as a small example E1 using f_1 (figure 10) and $n = 100$ (D2) for all other examples E2–E5 using $f_2, \ldots, f_4$ (figures 11 to 14). All test functions where sampled in challenging regions with interesting features. The affine coordinate transformations to the square domain and details on the experiments are summarized in table 2. Except for f_1 equation (17) we include the optional step of removing critical edges by edge flips.

In all plots, we color an edge red if it is a critical outflow edge and green if it is a critical inflow edge. Saddles are shown in yellow, maxima in red, and minima in blue.

Invertibility of integral lines We show the invertibility of integral lines by densely sampling integral lines over the complete domain of E1 (figure 10c). The integral lines can be arbitrarily close–but they never merge. Especially, their end points on the boundary are distinct (figure 10d). This can also be observed in the close-up views of E2 (figure 11c), E3 (figure 12c), and E4 (figure 13c).

Application in computing MS complexes Even though our method applies to integral lines of PL scalar fields in general, finding the MS complex and its separatrices is one major application of our method. As such, we demonstrate that our method results in PL MS complexes, where each cell is indeed a quadrangle with a well defined area like for continuous MS functions. Computing the separatrices is simple given a PL scalar field without critical edges: They can be computed by tracing integral lines starting at the local extrema of the slope function $a(t)$ of each saddle (figure 15).

Even for highly degenerate inputs with many spurious critical points such as E3 (figure 12), or E4 (figure 13) our method produces valid results. We compare our MS complex of E4 to the result of the Forman-based method used in ParaView [AGL05] via the Topology Toolkit (TTK) [SN12; MBF*21], which produces results which suffer from merging separatrices (figure 13d).

Errors in PL scalar fields Both, the input and the output of our algorithm are assumed to be sampled from smooth functions. This sampling already introduces an approximation error. We want to measure the error that our algorithm introduces and we compare PL input to PL output. (Alternatively, we could compute both with the sampled smooth function. However, typically this is unknown and available only for benchmarks.) We evaluate the PL input and output fields over a dense regular grid of 2000×2000 sample points (excluding points in cut holes) each and compute the absolute difference of the PL scalar fields. We determine the median and the maximum error and normalize by the magnitude of the scalar range to obtain relative errors, which are summarized in table 2.

Runtime We ran the examples on a desktop PC with an AMD Ryzen 9 3950X processor, 64GB DDR4 RAM, running GNU/Linux, and using Julia version 1.11.7 and one single core. All timings, except E1, include only the optional edge-flip preprocessing step and the runtime of our core algorithm. Table 2 summarizes results. All times are means from repeated runs.

6. Discussion and limitations

Extension to 3D Our method is limited to 2D PL scalar fields. For 3D PL scalar fields defined w.r.t. a tetrahedral partition of its

	function	domain	transformation	crit. edges	after flip	e-median	e-max	t-min	t-mean
E1	f_1 equation (17)	D1	$3\mathbf{x}+\mathbf{0}$	55	—	$1.479{\cdot}10^{-03}$	$4.138{\cdot}10^{-02}$	6.7 ms	7.2 ms
E2	f_2 equation (18)	D2	$2.6\mathbf{x}+\mathbf{0}$	614	109	$1.569{\cdot}10^{-09}$	$3.164{\cdot}10^{-03}$	214.6 ms	258.0 ms
E3	f_3 equation (19)	D2	$2.5\mathbf{x}+(0,1)^\top$	735	350	$3.648{\cdot}10^{-10}$	$7.536{\cdot}10^{-04}$	242.5 ms	265.2 ms
E4	f_4 equation (20)	D2	$9\mathbf{x}+\mathbf{0}$	1 940	1 034	$7.344{\cdot}10^{-10}$	$5.809{\cdot}10^{-03}$	342.4ms	376.5 ms
E5	f_4 equation (20)	D2	$20\mathbf{x}+\mathbf{0}$	3 722	2 540	$6.174{\cdot}10^{-10}$	$2.205{\cdot}10^{-02}$	557.8 ms	614.5 ms

Table 2: *Results of E1–E5. We measured the median and maximum of the sampled error (e) and the minimum and mean of the runtime (t).*

domain, the merging of integral lines occurs across triangles shared by adjacent tetrahedra. While this can be resolved by a vertex split similar to 2D, the algorithm is expected to be significantly more complex. However, we see no fundamental reason that prevents an extension to 3D. We leave the 3D case as subject of future research.

Integer arithmetic vs. combinatorial approach Existing discretizations of integral lines either rely on combinatorial approaches or on integer arithmetic. While both remove computational uncertainty, combinatorial methods are "better" in a sense that no growing, potentially arbitrarily large integer numbers are involved. While this would be a highly-desired improvement, combinatorial methods clearly cannot produce invertible integral lines.

Benefits for downstream applications We demonstrate the use of our algorithm by computing the Morse-Smale complex, which is drastically simplified by our method: Instead of restricting integral lines via different rules during integration like Bremer et al. [BP07], naive tracing of integral lines through triangles is now sufficient to produce valid results.

Integer arithmetic vs. floating-point arithmetic Our approach guarantees correct results when using integer arithmetic with arbitrary precision. While this gives a theoretical guarantee, the requirement for arbitrary precision is infeasible practice: Arbitrary precision can be "simulated" with libraries such as GMP. But this comes with a significant performance penalty. More importantly, the results cannot be simply "converted" to a standard floating-point representation. Instead, all downstream applications, which use the transformed PL scalar field, must use the same representation. The latter is impractical for any real-world application. For this reason, we stick to a floating-point implementation which brings benefits in performance but also the limitation that theoretical guarantees do not apply anymore. Figure 14 shows a challenging example, where we explore our algorithms limits with 64-bit floating-point precision such that in some local regions, critical edges cannot be resolved anymore: the numeric precision becomes insufficient for the representation of (nearly) degenerate triangles.

Triangle quality While our approach delivers PL fields with invertible integral lines, there are no guarantees on the quality of the resulting triangulation. This means that the resulting triangulation is valid but may contain poorly shaped–acute or obtuse–triangles, i.e., the minimal angle becomes small, and in extreme cases the triangle becomes near degenerate or collapses. We identify two reasons for extreme cases: Firstly, gradients almost parallel to edges: while condition A1 ensures that no triangle gradient is exactly parallel to an edge, "almost parallel" configurations may result in a valid triangulation with poorly shaped triangles. Secondly, cascading effects for chains of critical edges: chains of critical edges require a repeated application of local operations. This means that newly created triangles may undergo subsequent further modifications, each of which deteriorates the previous triangle shape. We attempt to minimize this effect by randomizing the processing order for local operations.

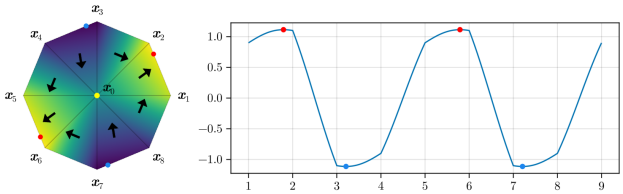

Figure 15: *(a): Saddle vertex of degree 8 with only inner local extrema in $a(t)$. (b): Slope function $a(t)$.*

Critical points While our approach requires a special treatment of local extrema by cutting the regions (section 4.1), no special treatment is necessary for saddle points: On termination of the algorithm, all saddle points have only inner local extrema in their corresponding slope function $a(t)$ (section 3.4). Figure 15 illustrates such a saddle vertex of degree 8. Removing higher-order saddles is necessary to obtain a proper Morse-Smale complex, but is not necessary for our algorithm.

Are we allowed to change the data? Our approach applies a (small) modification of the input data. This raises the question if this is allowed at all, or if this falsifies the data. We argue that the input PL field is already an approximation of an underlying unknown smooth field $\tilde{s}$. Our approach modifies this approximation and enforces properties of the smooth field. In this sense, our method brings the approximation even closer to the unknown original than the input PL field ever was. The same applies to resulting changes of the integral lines themselves. We argue that comparing any distance measure between integral lines p and p' starting at a point x in the input field s and output field s' is meaningless since p is not invertible and thus naturally differs from the unknown smooth integral line $\tilde{p}$ of the smooth field $\tilde{s}$.

Therefore, we only compare input and output scalar values (see section 5 and table 2) empirically: These experiments show that while our method removes critical edges, it still preserves the original input, i.e., it does not introduce significant errors or even generate "unrelated" scalar fields.

7. Conclusion

We identified critical edges as the reason for non-invertible integral lines and defined a local operator to resolve such edges. The repeated application of this operator transforms a piecewise linear scalar field into an arbitrarily close piecewise linear scalar field with invertible integral lines. In theory, our method always gives correct results for any valid input when using integer arithmetic with arbitrary precision. In practice, it performed reasonably well using floating-point arithmetic. While our results show that we are able to handle difficult inputs, triangles may degenerate due to lack of numerical precision.

Future work includes improving numerical stability in practice. One possibility could be to consider multiple critical edges, e.g., chains of critical edges, at once, to mitigate effects of repeated application of the local operator. Furthermore, an extension to 3D is of interest as well as a parallel implementation.

Code Availability

The code used in this work is available at https://visual2.cs.ovgu.de/timm.erxleben/iil2dplsf.

Acknowledgement

This work was partially supported by DFG grant TH 692/22-1. Open access funding enabled and organized by Projekt DEAL.

References

[AGL05] AHRENS, JAMES P., GEVECI, BERK, and LAW, C. CHARLES. "ParaView: An End-User Tool for Large-Data Visualization". *The Visualization Handbook*. Ed. by HANSEN, CHARLES D. and JOHNSON, CHRISTOPHER R. Academic Press / Elsevier, 2005, 717–731. DOI: 10.1016/B978-012387582-2/50038-1 10.

[BEHP04] BREMER, PEER-TIMO, EDELSBRUNNER, HERBERT, HAMANN, BERND, and PASCUCCI, VALERIO. "A Topological Hierarchy for Functions on Triangulated Surfaces". *IEEE Trans. Vis. Comput. Graph.* 10.4 (2004), 385–396. DOI: 10.1109/TVCG.2004.3 1, 3.

[BEKS17] BEZANSON, JEFF, EDELMAN, ALAN, KARPINSKI, STEFAN, and SHAH, VIRAL B. "Julia: A fresh approach to numerical computing". *SIAM Review* 59.1 (2017), 65–98. DOI: 10.1137/141000671 8.

[BHJ*14] BONNEAU, GEORGES-PIERRE, HEGE, HANS-CHRISTIAN, JOHNSON, CHRIS R., et al. "Overview and State-of-the-Art of Uncertainty Visualization". *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Ed. by HANSEN, CHARLES D., CHEN, MIN, JOHNSON, CHRISTOPHER R., et al. London: Springer London, 2014, 3–27. ISBN: 978-1-4471-6497-5. DOI: 10.1007/978-1-4471-6497-5_1 1.

[BJB*11] BHATIA, HARSH, JADHAV, SHREERAJ, BREMER, PEER-TIMO, et al. "Edge maps: Representing flow with bounded error". *2011 IEEE Pacific Visualization Symposium*. IEEE, Mar. 2011, 75–82. DOI: 10.1109/pacificvis.2011.5742375. URL: http://dx.doi.org/10.1109/PACIFICVIS.2011.5742375 2.

[BJB*12] BHATIA, HARSH, JADHAV, SHREERAJ, BREMER, PEER-TIMO, et al. "Flow Visualization with Quantified Spatial and Temporal Errors Using Edge Maps". *IEEE Transactions on Visualization and Computer Graphics* 18.9 (2012), 1383–1396. DOI: 10.1109/TVCG.2011.265 2.

[BKP*10] BOTSCH, MARIO, KOBBELT, LEIF, PAULY, MARK, et al. *Polygon Mesh Processing*. A K Peters, 2010. ISBN: 978-1-56881-426-1 4.

[BP07] BREMER, PEER-TIMO and PASCUCCI, VALERIO. "A Practical Approach to Two-Dimensional Scalar Topology". *Topology-based Methods in Visualization*. Ed. by HAUSER, HELWIG, HAGEN, HANS, and THEISEL, HOLGER. Mathematics and Visualization. Springer, 2007, 151–169. DOI: 10.1007/978-3-540-70823-0_11 3, 11.

[ČMD11] ČOMIĆ, LIDIJA, MESMOUDI, MOHAMMED MOSTEFA, and DE FLORIANI, LEILA. "Smale-Like Decomposition and Forman Theory for Discrete Scalar Fields". *Discrete Geometry for Computer Imagery*. Ed. by DEBLED-RENNESSON, ISABELLE, DOMENJOUD, ERIC, KERAUTRET, BERTRAND, and EVEN, PHILIPPE. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, 477–488. ISBN: 978-3-642-19867-0. DOI: 10.1007/978-3-642-19867-0_40 3.

[CSZ16] CAMPEN, MARCEL, SILVA, CLÁUDIO T., and ZORIN, DENIS. "Bijective maps from simplicial foliations". *ACM Transactions on Graphics* 35.4 (July 2016), 1–15. ISSN: 1557-7368. DOI: 10.1145/2897824.2925890. URL: http://dx.doi.org/10.1145/2897824.2925890 3, 8.

[DBG*06] DONG, SHEN, BREMER, PEER-TIMO, GARLAND, MICHAEL, et al. "Spectral surface quadrangulation". *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06. Boston, Massachusetts: Association for Computing Machinery, 2006, 1057–1066. ISBN: 1595933646. DOI: 10.1145/1179352.1141993 3.

[DK21] DANISCH, SIMON and KRUMBIEGEL, JULIUS. "Makie.jl: Flexible high-performance data visualization for Julia ". *Journal of Open Source Software* 6.65 (2021), 3349. DOI: 10.21105/joss.03349 8.

[DL12] DIECI, LUCA and LOPEZ, LUCIANO. "A survey of numerical methods for IVPs of ODEs with discontinuous right-hand side". *Journal of Computational and Applied Mathematics* 236.16 (2012), 3967–3991. DOI: 10.1016/j.cam.2012.02.011 2.

[DLG90] DYN, NIRA, LEVINE, DAVID, and GREGORY, JOHN A. "A butterfly subdivision scheme for surface interpolation with tension control". *ACM Trans. Graph.* 9.2 (1990), 160–169. DOI: 10.1145/78956.78958 5.

[EHNP03] EDELSBRUNNER, HERBERT, HARER, JOHN, NATARAJAN, VIJAY, and PASCUCCI, VALERIO. "Morse-smale complexes for piecewise linear 3-manifolds". *Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, CA, USA, June 8-10, 2003*. Ed. by FORTUNE, STEVEN. ACM, 2003, 361–370. DOI: 10.1145/777792.777846 3.

[EHZ03] EDELSBRUNNER, HERBERT, HARER, JOHN, and ZOMORODIAN, AFRA. "Hierarchical Morse - Smale Complexes for Piecewise Linear 2-Manifolds". *Discret. Comput. Geom.* 30.1 (2003), 87–107. DOI: 10.1007/S00454-003-2926-5 1, 2, 4.

[Fil88] FILIPPOV, ALEKSEJ F. *Differential Equations with Discontinuous Righthand Sides*. Springer Dordrecht, 1988. DOI: 10.1007/978-94-015-7793-9 2.

[For01] FORMAN, ROBIN. "A User's Guide To Discrete Morse Theory". *Sém. Lothar. Combin.* 48 (Dec. 2001) 1, 3.

[For98] FORMAN, ROBIN. "Morse Theory for Cell Complexes". *Advances in Mathematics* 134.1 (1998), 90–145. DOI: 10.1006/aima.1997.1650 3.

[GBHP08] GYULASSY, ATTILA, BREMER, PEER-TIMO, HAMANN, BERND, and PASCUCCI, VALERIO. "A Practical Approach to Morse-Smale Complex Computation: Scalability and Generality". *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), 1619–1626. DOI: 10.1109/TVCG.2008.110 3.

[GBP19] GYULASSY, ATTILA, BREMER, PEER-TIMO, and PASCUCCI, VALERIO. "Shared-Memory Parallel Computation of Morse-Smale Complexes with Improved Accuracy". *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), 1183–1192. DOI: 10.1109/TVCG.2018.2864848 2, 3.

[GGL*14] GYULASSY, ATTILA, GÜNTHER, DAVID, LEVINE, JOSHUA A., et al. "Conforming Morse-Smale Complexes". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 2595–2603. DOI: 10.1109/TVCG.2014.2346434 3.

[GS02] GOUZÉ, JEAN-LUC and SARI, TEWFIK. "A class of piecewise linear differential equations arising in biological models". *Dynamical systems* 17.4 (2002), 299–316. DOI: 10.1080/1468936021000041681 2.

[Man78] MANNSHARDT, REINHOLD. "One-step methods of any order for ordinary differential equations with discontinuous right-hand sides". *Numerische Mathematik* 31 (1978), 131–152. DOI: 10.1007/BF01397472 2.

[MB97] MALMBORG, JÖRGEN and BERNHARDSSON, BO. "Control and simulation of hybrid systems". *Nonlinear Analysis: Theory, Methods & Applications* 30.1 (1997), 337–347. ISSN: 0362-546X. DOI: 10.1016/S0362-546X(97)00048-5 2.

[MBF*21] MASOOD, TALHA BIN, BUDIN, JOSEPH, FALK, MARTIN, et al. "An Overview of the Topology ToolKit". *Topological Methods in Data Analysis and Visualization VI*. Ed. by HOTZ, INGRID, BIN MASOOD, TALHA, SADLO, FILIP, and TIERNY, JULIEN. Cham: Springer International Publishing, 2021, 327–342. ISBN: 978-3-030-83500-2. DOI: 10.1007/978-3-030-83500-2_16 2, 3, 10.

[MDS24] MIFTARI, EGZON, DURSTEWITZ, DANIEL, and SADLO, FILIP. "Visualization of Discontinuous Vector Field Topology". *IEEE Transactions on Visualization and Computer Graphics* 30.1 (2024), 45–54. DOI: 10.1109/TVCG.2023.3326519 2.

[MLP19] MANCINELLI, CLAUDIO, LIVESU, MARCO, and PUPPO, ENRICO. "A comparison of methods for gradient field estimation on simplicial meshes". *Comput. Graph.* 80 (2019), 37–50. DOI: 10.1016/J.CAG.2019.03.005 2.

[MLT*24] MAACK, ROBIN G. C., LUKASCZYK, JONAS, TIERNY, JULIEN, et al. "Parallel Computation of Piecewise Linear Morse-Smale Segmentations". *IEEE Trans. Vis. Comput. Graph.* 30.4 (2024), 1942–1955. DOI: 10.1109/TVCG.2023.3261981 3.

[MPZ14] MYLES, ASHISH, PIETRONI, NICO, and ZORIN, DENIS. "Robust field-aligned global parametrization". *ACM Trans. Graph.* 33.4 (2014), 135:1–135:14. DOI: 10.1145/2601097.2601154. URL: https://doi.org/10.1145/2601097.2601154 3.

[PK08] PIIROINEN, PETRI T. and KUZNETSOV, YURI A. "An event-driven method to simulate Filippov systems with accurate computing of sliding motions". *ACM Trans. Math. Softw.* 34.3 (May 2008). ISSN: 0098-3500. DOI: 10.1145/1356052.1356054 2.

[RGH*12] REININGHAUS, JAN, GÜNTHER, DAVID, HOTZ, INGRID, et al. "Combinatorial Gradient Fields for 2D Images with Empirically Convergent Separatrices". *CoRR* abs/1208.6523 (2012). arXiv: 1208.6523 10.

[Ros60] ROSENBROCK, H. H. "An Automatic Method for Finding the Greatest or Least Value of a Function". *The Computer Journal* 3.3 (Jan. 1960), 175–184. ISSN: 0010-4620. DOI: 10.1093/comjnl/3.3.175 10.

[RS14] RAY, NICOLAS and SOKOLOV, DMITRY. "Robust Polylines Tracing for N-Symmetry Direction Field on Triangulated Surfaces". *ACM Trans. Graph.* 33.3 (June 2014). ISSN: 0730-0301. DOI: 10.1145/2602145 2.

[SN12] SHIVASHANKAR, NITHIN and NATARAJAN, VIJAY. "Parallel Computation of 3D Morse-Smale Complexes". *Comput. Graph. Forum* 31.3pt1 (2012), 965–974. DOI: 10.1111/J.1467-8659.2012.03089.X 3, 10.

[SZ12] SZYMCZAK, ANDRZEJ and ZHANG, EUGENE. "Robust Morse Decompositions of Piecewise Constant Vector Fields". *IEEE Trans. Vis. Comput. Graph.* 18.6 (2012), 938–951. DOI: 10.1109/TVCG.2011.88 3.

[TAW24] THANH, SON LE, ANKELE, MICHAEL, and WEINKAUF, TINO. "Revisiting Accurate Geometry for Morse-Smale Complexes". *IEEE Topological Data Analysis and Visualization, TopoInVis@VIS 2024, St. Pete Beach, FL, USA, October 13-14, 2024*. IEEE, 2024, 34–43. DOI: 10.1109/TOPOINVIS64104.2024.00008 3, 9, 10.

[Van24] VANDENHEUVEL, DANIEL J. "DelaunayTriangulation.jl: A Julia package for Delaunay triangulations and Voronoi tessellations in the plane". *Journal of Open Source Software* 9.101 (Sept. 2024), 7174. DOI: 10.21105/joss.07174 10.

Appendix A

Proof of Lemma 1 To study the occurrence of local vertex extrema of a at the vertex i, we consider the first derivative $\dot{a}(t)$ at $t = i$. Since $\dot{a}(t)$ is discontinuous at $t = i$, we need to approach $t = i$ from left and from right denoted $\dot{a}_-(t)$ and $\dot{a}_+(t)$, respectively. Elementary computations yield

$$\dot{a}_-(i) = \frac{2A_{0,i-1,i}}{\|\mathbf{h}_{0i}\|} q_{i0} \quad \text{and} \quad \dot{a}_+(i) = -\frac{2A_{0,i,i+1}}{\|\mathbf{h}_{0i}\|} q_{0i}. \tag{21}$$

The slope $a(t)$ has a local vertex extremum at $t = i$ iff $\dot{a}_-(i)$ and $\dot{a}_+(i)$ have opposite sign. Since, $A_{0,i-1,i} > 0$, $A_{0,i,i+1} > 0$, and $\|\mathbf{h}_{0i}\| > 0$, this requires q_{i0} and q_{0i} to have equal sign. Inserting definition 1 gives that the edge $\{0,i\}$ is critical. □

Proof of Theorem 1 If the field has no critical edges, slope functions do not have any local vertex extrema. Therefore, global extrema of $a(t)$ are guaranteed to be inner extrema. In this case, the integral line through $\mathbf{x}_0$ passes the interior of a triangle rather than an edge, meaning that no merging or splitting of the integral line can occur at the center vertex 0. This, together with integral line behavior across edges as discussed in section 3.3, means that merging or splitting of integral lines is solely due to critical edges, regardless of whether the line crosses an edge or a vertex. □

Proof of Lemma 2 We have to show $q_{jk} < 0$. W.l.o.g. we assume

$$\mathbf{x}_i = (0,0) \quad , \quad \mathbf{x}_j = (x,y) \quad , \quad \mathbf{x}_k = (-x,y) \tag{22}$$

with $x, y > 0$. Inserting this in equation (2) and equation (3) gives

$$q_{ij} = -\frac{(2s_i - s_j - s_k)x}{y}. \tag{23}$$

This, $y > 0$ and $s_i > s_j, s_k$ give $q_{ij} < 0$ which proves the lemma. □

Appendix B

Additional supporting information may be found online in the Supporting Information section at the end of the article.