Tracking vortices in superconductors: Extracting singularities from a discretized complex scalar field evolving in time

Carolyn L. Phillips, Hanqi Guo, Tom Peterka, Dmitry Karpeyev, and Andreas Glatz

Phys. Rev. E **93**, 023305 — Published 19 February 2016

DOI: 10.1103/PhysRevE.93.023305

Tracking vortices in superconductors: Extracting singularities from a discretized complex scalar field evolving in time

Carolyn L. Phillips,[1,*] Hanqi Guo,[1,†] Tom Peterka,[1] Dmitry Karpeyev,[1] and Andreas Glatz[2]

[1]*Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA*
[2]*Materials Science Division, Argonne National Laboratory, Lemont, IL 60439, USA*

(Dated: January 29, 2016)

In type-II superconductors, the dynamics of magnetic flux vortices determine their transport properties. In the Ginzburg-Landau theory, vortices correspond to topological defects in the complex order parameter field. Earlier, in [Phillips, et al. PRE 91 (2), 023311] we introduced a method for extracting vortices from the discretized complex order parameter field generated by a large-scale simulation of vortex matter. With this method, at a fixed time step, each vortex (simplistically, a 1D curve in 3D space) can be represented as a connected graph extracted from the discretized field. Here we extend this method as a function of time as well. A vortex now corresponds to a 2D space-time sheet embedded in 4D space-time that can be represented as a connected graph extracted from the discretized field over both space and time. Vortices that interact by merging or splitting correspond to disappearance and appearance of holes in the connected graph in the time direction. This method of tracking vortices, which makes no assumptions about the scale or behavior of the vortices, can track the vortices with a resolution as good as the discretization of the temporally evolving complex scalar field. Additionally, even details of the trajectory between time steps can be reconstructed from the connected graph. With this form of vortex tracking, the details of vortex dynamics in a model of a superconducting materials can be understood in greater detail than previously possible.

I. INTRODUCTION

Many phenomena in nature can be described by the behavior of complex scalar functions or vector fields, ranging from electromagnetic fields to director fields in liquid crystals, spins in magnets, and complex order parameters in superfluids and superconductors. Topological defects in those functions or fields represent important features of the underlying physical system: Examples are (zero-dimensional) point defects or monopoles, (one-dimensional) defect lines or strings, and (two-dimensional) domain walls. Here we concentrate on defect lines, which in the case of a complex scalar field are defined by one-dimensional manifolds, where the phase of the complex function is undefined. These topological singularities or defects are typically associated with circulations in the phase gradient and are referred to simply as vortices. Substantial work has been invested in studying the dynamics of vortices in different contexts, such as crossing and reconnection and the formation of knots in superfluid vortices [1, 2], in light waves [3], and in fluid flows [4], as well as their evolution in more mathematically generalized contexts [5].

In type-II superconductors, an externally applied magnetic field penetrates the system above the first critical field in the form of flux tubes (vortices), which carry integer numbers of flux quanta (typically one flux quantum). The magnetic flux in the vortex core is screened by a circular superconducting current around it. The behavior of vortices carrying magnetic flux determines the material's ability to sustain the dissipationless or superconducting state. When vortices move, the system becomes dissipative, and a finite voltage drop across the system is observed. In the Ginzburg-Landau theory of superconductivity, the local superconducting properties of the material are described by a spatially dependent complex order parameter ψ, and vortices correspond to topological phase singularities of ψ accompanied by a suppression of its magnitude. Using the time-dependent Ginzburg-Landau (TDGL) equations, coupled partial differential equations evolving the scalar ψ field in time, one can find steady-state solutions of the superconductor in the presence of external magnetic fields and applied currents.

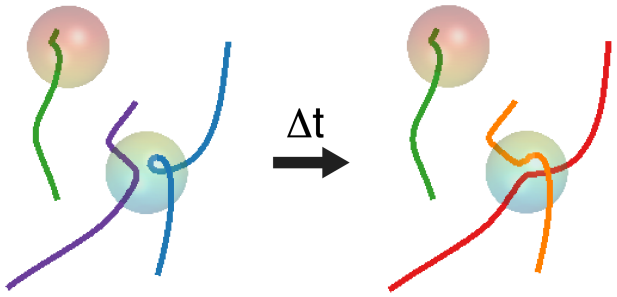

FIG. 1: (Color online) Two time-step images of three vortices interacting with two pinning sites. The upper left vortex experiences no events across the time interval. The bottom two vortices cut and reconnect or merge and split into two new vortices.

In the dissipative regions of a superconductor, vortices are dynamic objects that can nucleate and annihilate; they can also cut each other and reconnect (Fig. 1). In static situations, vortices can be pinned by material defects inside the superconductor. Recently, to study the collective dynamics of many vortices, researchers have begun conducting large, computationally intensive 3D simulations where macroscale phenomena can be observed [6, 7]. These simulation have been used to examine the dynamics of flux cutting, where two vortices move through each other [8–10], to study the impact of distri-

* corresponding author *E-mail address:* cphillips@anl.gov
† CLP and HG contributed equally to this work

butions of defects, and to optimize the pinning of vortices by spherical nano particles [11].

As the scale of simulations increases, visualizing and quantifying the behavior of a large collection of vortices requires the codesign of analysis techniques that can scale with the application and even improve the resolution with which dynamics are observed. In Ref. [12], a method for extracting the topological defect lines from a large data set of complex scalars defined over a mesh at a single time step was introduced that permits details of vortex interactions to be understood in fine detail. In order to understand the relationship between dissipation and the dynamics of the vortices, however, the details of vortex interactions need to be understood over time as well as space. Specifically, in order to describe an event, such as two vortices recombining, the individual vortices participating in the event and isolated at an initial time step must be tracked over subsequent time steps.

Here we show how the method for numerically extracting a vortex from a complex order parameter field can be extended to work over time as well as space. This analysis is parameterless and makes no assumptions about the shape, velocity, or behavior of a vortex. It assumes only that the simulation data changes smoothly as a function of time. As such, this method represents the highest resolution interpretation of the identity and dynamics of a vortex over time that is possible given the resolution of the simulation data set.

This analysis has applications to data sets of discretized complex fields containing topological defects as long as the field evolves smoothly in time. Examples of complex fields containing topological defects include optical vortices in electromagnetic fields as well as other problems described by the complex Ginzburg-Landau equations such as screw dislocations [13] cosmic strings [14], superfluidity, and Bose-Einstein condensation; strings in field theory [15]; topological defects in liquid crystals [16]; and models of fluid dynamics with complicated nonlinear dynamics [17].

In Section II we provide background information on vortex extraction and tracking. In Section III we first review how topological defects are extracted as a graph structure from a field discretized over space, and we then extend this method to a space-time field. In Section IV we discuss how events can be interpreted from the behavior of this graph structure. In Section V we show how this graph extraction can be implemented in an algorithm that is a simple extension of the algorithm presented for extracting a vortex from a spatial mesh, and we consider the scaling of the algorithm. In Section VI we present examples of this algorithm applied to simulation data. In Section VII, we provide concluding remarks about the current and potential benefits of the method.

II. BACKGROUND

Feature tracking is widely studied in various computer science research directions, such as computer vision, image processing, and visualization, and is often applied to scientific data sets. In general, the definition of a feature depends on the nature of the data. Most tracking methods are based on a correspondence analysis that determines what feature in one time frame corresponds to what feature in the next frame [38]. For example, feature correspondence can be determined by the overlap of feature volumes or by attribute similarities in adjacent frames. In some data sets, features can also appear, disappear, or interact with each other over time.

For type-II superconductors, the feature of interest are the magnetic flux vortices. In numerical studies of type-II superconductors, many different model types have been proposed to capture vortex interactions. By treating the vortices as interacting elastic strings [18, 19], it is possible to study the equilibrium states [20–22] and dynamics [23, 24] of many vortices. Tracking vortices modeled as elastic strings is trivial, as the set of vortices are explicit objects being evolved by the simulation. However, such systems treat interactions between vortices and interactions between vortices and pinning sites only approximately and cannot capture vortex cutting and reconnection, and therefore are only suitable for small magnetic fields.

By instead modeling the dynamics of a superconducting order parameter ψ, where vortices appear spontaneously as singularities of the order parameter, more realistic descriptions of vortex matter can be constructed [25–33]. Now defined as topological defect lines in the field, vortices are implicit features in the data that need to be extracted. Vortices can be identified by examining the contour plots of $|\psi|$ in 2D [34–36] (or the isosurfaces of $|\psi|$ in 3D [37]). Alternately, vortices can be extracted by using discretized contour integrals to find defect points in the phase field of the order parameter. For 3D data sets, the points can be linked together to form vortex lines [12, 28–31]. As the scale of simulations increase [6], the details of the implementation of the vortex extraction algorithm and the final representation of a set of vortices become important considerations for vortex extraction to keep pace with vortex simulation [12]. From here forward in this article, all references to vortices in a superconductor will assume they are singularities of an order parameter defined over a field, not elastic strings.

Tracking vortices is only meaningful for simulations that reproduce dynamics. It is not relevant, for example, to simulations that use a Monte Carlo method for relaxing the order parameter [27–31] that generate data sets of statistically independent 3D snapshots that can be arranged in any order. In contrast, the data sets generated by TDGL simulations are proper 4D data sets describing vortex matter evolving over time. To investigate vortex dynamics in such a data set, vortices need to be extracted from 3D slices and then also tracked over the time dimension of the data set. Using a correspondence analysis based on attribute similarity is not applicable to these simulations as vortices have no attributes, (e.g. shape, length) that are guaranteed to be stable over time (except chirality). In fact, a vortex only has a conserved identity between its birth, death, and interactions with other vortices.

Vortex tracking in a superconductor is closely related to vortex tracking in fluid flows, which is an established topic in scientific visualization [39]. Whereas magnetic flux vortices have a single mathematical definition, fluid vortices can be characterized by multiple criterion, such as vorticity mag-

nitude and λ_2 [40]. For different applications and definitions, vortices in fluids are extracted and tracked as vortex regions or vortex core lines. Vortex regions can be located by applying a criteria with a threshold to a data set, and vortex core lines can be extracted and tracked with techniques such as the parallel vector operators [41] and feature flow fields [42] frameworks, respectively.

In this paper, we propose a graph-based algorithm for tracking vortices in a superconductor at the same resolution as the data discretization. Compared with the methods used for fluid flows, this method provides a more straightforward and efficient way to track vortices in a superconductor by taking advantage of the structure and easily extracted local properties of the data. Earlier [12], we demonstrated how, by exploiting the mathematical definition of topological singularity, a vortex can be traced in space. Here we show that by extending our definition of a vortex as a topological singularity in a field that is continuous over space and time, a vortex can be extracted from the field as a quantized object defined in space and time. Thus a vortex can be traced over time as well. This method represents the most information about the vortex structure that can be extracted directly from the simulation data, against which any computationally cheaper method using approximations must be compared.

III. TOPOLOGICAL SINGULARITIES EVOLVING OVER TIME

In this section, we first review how topological singularities are extracted as a graph structure from a spatial mesh. We then extend the method to a space-time mesh.

A. Topological singularities in a spatial mesh

The spatially and temporally discretized field that we consider here results from TDGL equations solved over a structured or unstructured mesh, as introduced in Ref. [6]. The TDGL equations solve for the complex-valued order parameter $\psi = |\psi| e^{\imath\theta}$, and vortices are equivalent to topological singularities in the phase field of θ. Given a set of complex values ψ that have been calculated at each point of a mesh, vortex lines can be localized by calculating the integral

$$n = -\tfrac{1}{2\pi} \oint_{\mathscr{C}} \nabla\theta \cdot dl \tag{1}$$

around the closed contour $\mathscr{C}$. Assuming that the closed path is sufficiently small so as not to enclose multiple vortices, then when the vorticity n is a nonzero integer (usually ± 1), the path encircles a vortex line, and the sign of n indicates the chirality of the vortex with respect to the direction of integration. In general, we will construct these closed paths around mesh element faces.

For this analysis, a convenient way to represent the mesh is as its *dual graph*. In the dual graph, mesh elements are nodes, and an edge connects two nodes if the corresponding mesh elements share a face. For illustration, the nodes of this dual graph can be located in the center of each mesh element. In a structured mesh of hexahedral elements, which for simplicity we will refer to as cubes, each node has six undirected edges, one to each of six neighboring nodes. A vortex, then, embedded in the field described over a mesh corresponds to a set of punctured mesh element faces, that is, faces whose contour integral per Eq. (1) has a nonzero value. Equivalently, a vortex can be described as a set of nodes and directed edges that constitute a connected subgraph of the dual graph of the mesh. The direction of an edge in this subgraph is determined by and corresponds to the chirality of the vortex. Since most – indeed, nearly all – of the nodes of the subgraph have connectivity two, the one-dimensional curve describing a given vortex can be constructed by beginning at one node and tracing through the graph, generating an ordered set of spatial points. Rare nodes can have connectivity greater than two; these nodes correspond to locations in the mesh where multiple vortices puncture the same mesh element. By using chirality information, vortices can be disentangled in a nonunique way to support subsequent statistical analyses. We have found that by dividing a hexahedral element into tetrahedral subelements, these vortices can be uniquely disentangled. For computational convenience, however, such vortices may be left in an unresolved connected state.

In Ref. [12], a vortex object was defined as a reduced mathematical representation of a set of one-dimensional curves that usually corresponds to individual vortices, and less commonly two or more entangled vortices, in a discretized complex scalar field. Constructing a vortex object requires only one constraint on the data: at most only one vortex can puncture a given mesh element face. In general, the length scales of the mesh in a well-behaved simulation have already been selected so that this constraint should hold. However, if a coarser description of the mesh than the simulation mesh is used to trace the vortices – for example, a closed path is constructed around multiple mesh faces – then this can be a problem.

This graph representation of a vortex (or multiple entangled vortices) works equally well for both structured and unstructured meshes. However, for reasons of algorithmic efficiency related to how unstructured mesh libraries store mesh data, sometimes it is more convenient to transform the directed subgraph even further into its corresponding *line digraph*, or a *edge-to-vertex dual*, as discussed in Ref. [43], and perform all analysis in this paradigm instead. In this new graph interpretation, edges of the directed subgraph, which were punctured faces of the mesh, are reinterpreted as nodes; and nodes of the directed subgraph, which were punctured mesh elements, are split into a set of directed edges, one for each path through the mesh element. For the purpose of this paper, we stay in the directed subgraph paradigm and not the edge-to-vertex dual paradigm.

B. Topological singularities in a space-time mesh

In the TDGL equations, ψ is evolved over time and is therefore both a spatially and temporally dependent field, dis-

cretized spatially over the mesh and temporally over time steps. A smooth evolution of ψ over time leads to the generation of a topological defects in planes oriented in time as well as in space, which is discussed in more depth in the appendix of Ref. [43]. In the model for a superconducting material described in Ref. [6], the magnetic field and applied superconducting current can also be varied over time. For a small reformulation of Eq. (1) to account for the varying fields, topological defects can be detected in closed paths around planes oriented in time as well. Details are provided in Appendix B.

To trace a vortex in time, we now consider a structured mesh defined over time as well as space. The discretized temporal evolution of a spatially discretized field defined over a 2D or 3D structured mesh corresponds to a 3D or 4D space-time structured mesh, respectively. We start by describing the 2D structured mesh in space corresponding to 3D structured space-time mesh initially, where the properties of the mesh and the graph derived from the mesh are easier to visualize.

1. 2D structured spatial mesh, 3D structured space-time mesh

For a 2D structured spatial mesh, the space-time mesh is composed of three-dimensional cubic mesh elements. Each cube corresponds to a face in the 2D spatial mesh extended over a time interval between two time steps. Each cube has six neighboring cubes. It shares a face with each neighbor. These faces can be space faces, that is, a face defined by four spatial coordinates at a fixed time, or time faces, that is, a face that corresponds to a one edge in the 2D spatial mesh extended over the time interval. A 3D space-time mesh element is connected to a space-time mesh element over the previous time interval but at the same location in space by a space face. A 3D space-time mesh element is connected to a space-time mesh element over the same time interval but neighboring in space by a time face.

We can speak of a 3D space-time mesh element as being punctured by a topological defect, which is now a one-dimensional space-time curve. A 3D structured space-time mesh element punctured by an isolated topological defect, that is, one not entangled with another topological defect inside the mesh element, has two punctured faces.

In the dual graph of this 3D space-time mesh, nodes correspond to space-time mesh elements. For illustration, we place the nodes of this dual graph in the center of the 3D space-time mesh element. Each node in the dual graph has six edges, corresponding to six faces connecting it to its six neighboring 3D mesh elements. A vortex evolving over time is a subgraph of this dual graph. A vortex is a connected set of nodes and edges, where nodes correspond to punctured mesh elements and edges correspond to punctured faces. The connected set of nodes and edges is constructed by testing space and time faces to see whether they are punctured. Each punctured face will "activate" a single edge of this vortex subgraph. Only nodes with connectivity > 0 are part of the subgraph.

Unlike the subgraph described in section III A for a 3D spatial mesh, the edges in this graph are treated as undirected. The direction of an edge for the subgraph of a 3D spatial mesh comes from the chirality of the puncture point, as determined by whether $n = \pm 1$ relative to the direction of the contour integral. For these subgraphs, chirality information can be used to disambiguate the vortex structures inside the rare punctured cube with more than two punctured faces. While the chirality of the puncture point for a time face is still well defined, the chirality information provides marginal extra information. For the tracking method described here, this chirality information is not used in constructing the connected set of nodes and edges or in any subsequent analysis.

Figures 2 and 3 illustrate how a time evolution of a point vortex in a 2D mesh corresponds to a space-time curve embedded in a 3D space-time mesh, which can be represented as a subgraph of the dual graph of the 3D space-time mesh. We use the following notation to label nodes in the graph. If a node corresponds to the 3D mesh element that spans the interval (n_x, n_x+1), (n_y, n_y+1), and (n_t, n_t+1), where n_x, n_y, n_t are the discretized position and time coordinates of a mesh point, then the corresponding node of the dual graph has the label $[n_x, n_y, n_t]$.

In Fig. 2(a) a point vortex stays inside a mesh face from $t = 0$ to $t = 1$. Its trajectory through the 3D space-time mesh element is shown in Fig. 2(b). Two space faces, the top and bottom of the mesh element, are punctured. In Fig. 2(c) a dual graph is shown for the mesh element. The node [0,0,0] is connected by edges to its six neighbors. In Fig. 2(d), the subgraph of the dual graph is shown. Only edges corresponding to punctured faces and nodes that have at least connectivity one are contained in the dual graph. Edges are labeled by the label of their corresponding punctured faces.

In Fig. 3(a) a point vortex exits the mesh face at some time between $t = 0$ and $t = 1$ and enters a neighboring face. Its trajectory through the 3D space-time mesh elements is shown in Fig. 3(b). Now two space faces and one time face are punctured. Figure 3(c) shows the resultant subgraph of the dual graph (not shown). The three edges of the subgraph correspond to the three punctured faces of the 3D space-time mesh elements.

2. 3D structured spatial mesh, 4D structured space-time mesh

For a 3D structured spatial mesh, the space-time mesh is composed of four-dimensional hypercubic mesh elements. Each hypercube corresponds to a cube in the 3D spatial mesh extended over a time interval between two time steps. Each hypercube has eight neighboring hypercubes. It shares a volume, or cube, with each neighbor. The faces of these cubes can be space faces or time faces. A space-time mesh element is connected to the space-time mesh element at the same location in space but spanning the prior time interval by a cube with six space faces. This cube is defined at the time step they share and therefore corresponds to a mesh element of a 3D structured mesh at a single time step. A space-time mesh element is connected to a space-time mesh element over the same time interval but neighboring in space by a space-time cube. This cube has two space faces, which is the same face at two bounding time steps, and four time faces, correspond-

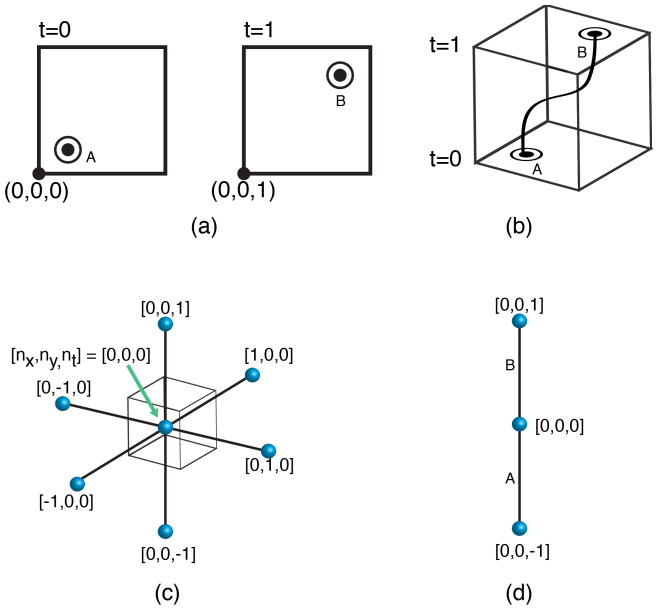

FIG. 2: (Color online) (a) Single mesh element of a 2D mesh punctured at t=0 and t=1. (b) Trajectory of the (point) vortex shown as a curve inside the single 3D space-time mesh element. (c) Dual graph of the mesh element. (d) Subgraph generated by puncture points A and B. Only space faces were punctured.

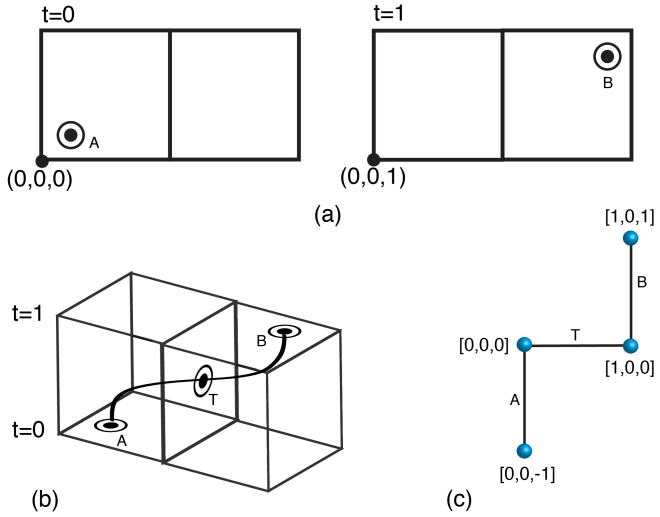

FIG. 3: (Color online) (a) Two mesh elements of a 2D mesh. The left element is punctured at t=0, and the right element is punctured at t=1. (b) Trajectory of the (point) vortex shown as a curve inside the two 3D space-time mesh element. A time face is punctured. (c) Subgraph generated by puncture points A, B, and T.

ing to the four edges of the space face extended over the time interval.

Every face is shared by four cubes. A space face is shared by two space cubes, a space-time cube spanning the previous time interval, and a space-time cube spanning the subsequent time interval. A time face is shared by four space-time cubes spanning the same time interval, corresponding to the four mesh elements of the spatial mesh that share the spatial edge. We note that if two 4D mesh elements share only a single face and not a cube, they are not neighbors, according to our definition, just as two diagonal 3D mesh elements that share an edge but not a face are not neighbors.

As before, we can speak of a 4D space-time mesh element as being punctured by a topological defect, which is now a two-dimensional space-time sheet. A 3D structured mesh element punctured by an isolated topological defect, that is, one not entangled with another topological defect inside the mesh element, has two punctured faces. Likewise a 4D mesh element punctured by an isolated topological defect has four punctured cubes. Again, each punctured cube has two punctured faces; that is, the vortex enters and exits each cube. Each punctured face is shared by two of the cubes.

In the dual graph of this 4D mesh, nodes correspond to space-time mesh elements. If a node corresponds to the 4D mesh element that spans the interval (n_x, n_x+1), (n_y, n_y+1), (n_z, n_z+1) and (n_t, n_t+1), where n_x, n_y, n_z, n_t are the discretized position and time coordinates of a mesh point, then the corresponding node of the dual graph has the label $[n_x, n_y, n_z, n_t]$. Each node in the dual graph has eight edges, corresponding to eight cubic volumes connecting it to its eight neighboring 4D mesh elements. It will be useful to think of each edge of this dual graph as corresponding to the bundle of six faces corresponding to the connecting cube, as illustrated in Fig. 4.

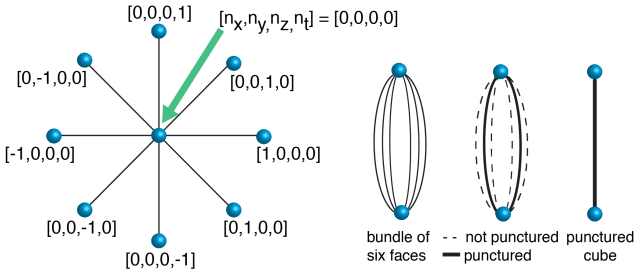

FIG. 4: (Color online) A node in the dual graph of the 4D space-time mesh connected to 8 neighbors. The indices of the node indicate its position in discretized space and time and the indices of its spatial and temporal neighbors. Each edge corresponds to a connecting cube that corresponds to a bundle of faces. In the vortex subgraph, two of the faces are punctured. The two punctured faces create a punctured cube, which creates a single undirected edge connecting the two nodes.

Again, a vortex evolving over time corresponds to a subgraph of this dual graph. A vortex is a connected set of nodes and edges, where nodes correspond to punctured 4D mesh elements and edges correspond to punctured cubes. As before, the connected set of nodes and edges is constructed by testing space and time faces to see whether they are punctured. However, now each punctured face indicates four punctured cubes

and thus will "activate" four separate edges of the vortex subgraph.

In general, the nodes of a subgraph that correspond to a vortex have connectivity greater than or equal to two. For example, a spatial vortex can have end points only at the non-periodic boundaries of the system. A point vortex in a 2D space evolving over time can have an end point only when it is born or dies. In a 4D mesh, almost all punctured cubes contain two punctured faces, one corresponding to in and a second corresponding to out. However, the analysis described here works equally well for data where the topological defect described has end points (e.g., birth and death or spatial end points not at boundaries).

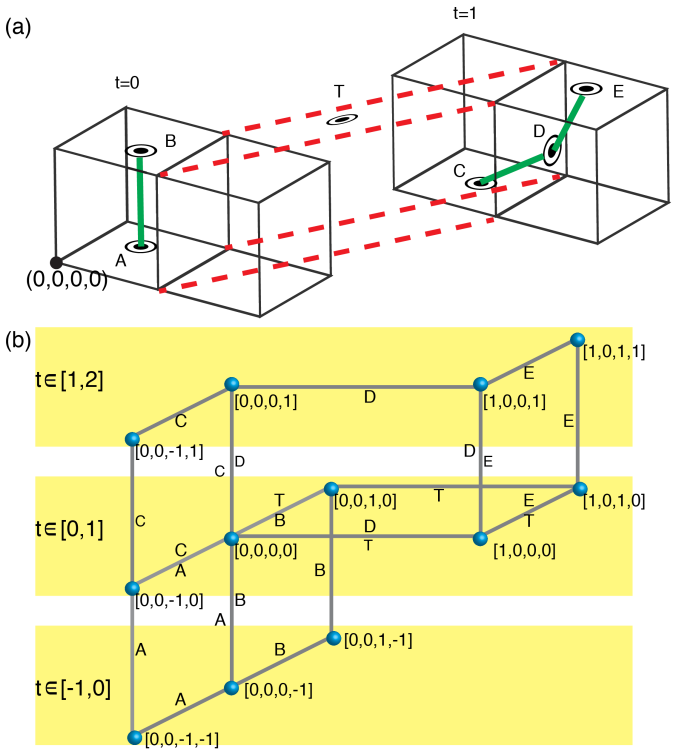

FIG. 5: (Color online) (a) Top part of a vortex moving over one face from t=0 to t=1. The punctured faces are labeled A, B, C, D, E, and T. (b) Resultant vortex graph structure. Nodes in each highlighted region correspond to the same time interval. Each edge corresponds to a punctured cube and is labeled by the puncture faces of the cube. The graph structure at t $\in [0,1]$ is connected to the graph structure at t $\in [-1,0]$ and to the graph structure at t$\in [1,2]$ by the vertical edges.

Analogous to the spatially defined mesh, tracing a vortex object over time requires only one constraint on the data: at most only one vortex can puncture a given mesh element time face or space face. In general, both the discretization of the length scale and the time scale of data from a well-behaved simulation should be such that this constraint should hold. However, if a coarser length scale or time step than the simulation length scale or time step is used when analyzing the data, then this can be a problem. Also, if a single space-time mesh element is punctured by more than one vortex space-time sheet, we do not disentangle them. The two vortex sheets are considered connected within our ability to resolve them.

Figures 5 and 6 illustrate how a time evolution of a line vortex in a 3D mesh corresponds to a space-time sheet embedded in a 4D space time mesh, which can be represented as a subgraph of the dual graph of the 4D space-time mesh. In Fig. 5, a section of a line vortex is initially in a single 3D spatial mesh element. From $t = 0$ to $t = 1$, the top half of the vortex slides over to a neighboring 3D spatial mesh element. As a result, five space faces (A, B, C, D, and E) and one time face (T) are punctured. Dashed lines assist visualization of the time faces. At the bottom of Figure 5, the activated nodes and edges of the dual graph are shown. Each edge is labeled by the punctured face or faces that activated it. Parts of the graph that correspond to the time interval $t \in$ [-1,0], [0,1], and [1,2] are highlighted.

In Fig. 6, we show how the constructed graph can identify that a vortex at $t = 0$ is the same as the vortex at $t = 1$, despite the two vortices sharing no common punctured face in the two 3D meshes. In Fig. 6(a), a section of a line vortex moves through multiple 3D spatial mesh elements from $t = 0$ to $t = 1$. The punctured time faces due to its movement are shown in Fig. 6(b). As a result of the movement, four space faces (A, B, C, and D) and four time faces (T1, T2, T3, and T4) are punctured. Nodes and edges are labeled the same as in Fig. 5; again, parts of the graph that correspond to common time intervals are highlighted. The nodes and edges activated by the T1, T2, T3, and T4 punctured faces connect the nodes and edges activated by the space faces. Thus the vortices embedded in the 3D mesh $t = 0$ and $t = 1$ correspond to a single connected subgraph of the dual graph of the 4D mesh, and therefore are the same vortex.

Our analysis relies on two assumptions. First, the contour integral performed over mesh faces correctly identifies punctured faces; that is, edges are correctly identified. For example, the contour integral calculation can produce an incorrect result if two vortices puncture the same face or if the phase changes by more than π along an edge of mesh. Second, the analysis assumes that two distinct vortices do not cross through the same region over the elapsed time interval. Both these assumptions typically hold if the analysis is based on the smallest space and time intervals available from the discretization of the simulation data. However, these assumptions can also hold for significant coarsening of the data if the vortices are dilute and move slowly.

In Appendix A, we show how the dual graph paradigm can be converted to the line digraph paradigm, using the graphs of Figs. 5 and 6 as examples.

IV. INTERPRETING EVENTS FROM ANALYSIS OF THE SPACE-TIME VORTEX GRAPH

Interpreting events does not require storing and then analyzing the entire space-time graph generated over an entire simulation. Rather it involves interpreting how the structure of the graph evolves from time step to time step.

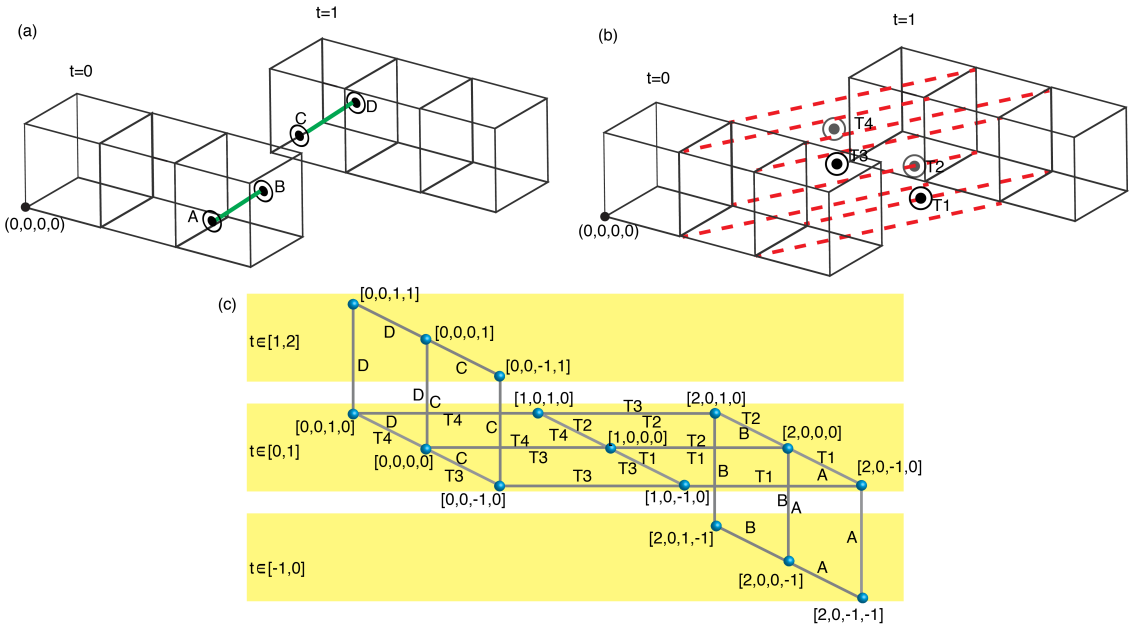

FIG. 6: (Color online) (a) Vortex moving multiple cells from right to left between t=0 and t=1. (b) Four punctured time faces (T1, T2, T3, and T4) resulting from the movement. (c) Connected graph structure at t$\in [0,1]$, which is connected to the graph structure at t$\in [-1,0]$ and t$\in [1,2]$ by vertical edges. Each edge corresponds to a punctured cube and is labeled by the punctured faces of the cube.

Given that individual vortices correspond to connected graphs in the subgraph, we now can define the following events.

- Continuity – (a nonevent) A single vortex at one time step is mapped to a single vortex in another. This corresponds to a connected graph at one time interval that is connected to a single connected graph at a subsequent time interval.

- Birth – A vortex in a time step was not present in the previous time step. It has spontaneously appeared, for example, emitted from a boundary. This corresponds to a connected graph at one time interval that is not connected to any connected graph in the prior time interval.

- Death – A vortex in a time step is not present in the subsequent time step. It has, for example, been absorbed into a boundary. This corresponds to a connected graph at a time interval that is not connected to any connected graph in the subsequent time interval.

- Merge – Two or more vortices join into a single structure. This corresponds to two (or more) connected graphs at one time interval that are connected to a single connected graph in the subsequent time interval.

- Split – A vortex breaks into two or more vortices. This corresponds to a connected graph at one time interval that is connected to two (or more) connected graphs at a subsequent time interval.

More complex events can occur by combining the events above.

- Birth-Split (Pair Production) – Two vortices of opposite chirality spontaneously appear. This corresponds to a connected subgraph at a time interval that is connected to no subgraph in the prior time interval and two subgraphs in the subsequent time interval.

- Merge-Death (Annihilation) – Two vortices of opposite chirality can annihilate each other in a simulation. This will appear as a merge followed by a death. An annihilation is equivalent to a pair production reversed in time.

- Merge-Split (Vortex Cutting/Recombination) – Two vortices can approach each other, touch at a point, and then separate. This may be an event where one vortex can be interpreted as moving through the other (cutting, crossing, and then reconnecting) or an event where the two vortices swap parts when they touch (recombination, as observed in Refs. [8–10]). Both events will appear as a merge followed by a split.

With respect to a merge-split, graph analysis by itself cannot distinguish a cut/reconnect from a recombination. To tell one event from the other, one must analyze the most likely path individual points along each vortices have traced through. This analysis requires choosing a mapping function to map individual points on a vortex in one time step to the points on the same vortex in the subsequent time step, and then assessing whether the points associated with a given vortex before the merge/split have mapped onto a single vortex after

the merge/split or have distributed across two vortices. While such a mapping function could be built on an analysis of the connected graph structures (e.g., shortest graph path connecting a space face puncture point in one time step to a space face puncture point in the subsequent time step), other mapping functions based on, for example, connecting points with the shortest displacements or using normals to the curve are equally legitimate and yet may map points to slightly different subsequent points.

In simulations with periodic boundary conditions, the interpretation of events can depend on how one divides a vortex with respect to the boundary conditions. The periodic boundary condition allows a single vortex to wrap through the simulation box multiple times, and each wrapping is commonly interpreted as an independent vortex. The interpretation of an event can depend on how the wrapped vortex is divided into pieces. For example, if one wrapping of a vortex merges with another wrapping, this will be interpreted as continuity, not a merge, unless different wrappings of the vortex are treated as independent vortices.

Here, we are interested only in events that correspond to changes in the connected components of the space-time graph. That is, the method described here does not detect events such as the pinning of a vortex on an inclusion or the change in internal topology of a vortex such as the splintering of a giant vortex into multiple vortices [44]). Detecting these types of events requires additional analysis of individually tracked vortices.

V. ALGORITHM

In this section we describe an algorithm for constructing, traversing, and interpreting the graph of Section IV that can be easily integrated with the algorithm described in Ref. [12] for extracting the vortex objects from a single time step.

Interpreting the events that occur across a time interval does not require constructing a full time-interval subgraph based on a 4D mesh; it can be achieved by constructing and performing operations on three subgraphs: (1) the subgraph for the 3D spatial mesh at time step (2) the subgraph for the 3D spatial mesh at the next spatial time step, and (3) the subgraph for the time interval mesh connecting the two time steps. Since the last mesh contains only one time interval, it can be projected into a 3D mesh. Thus, only minimal extensions to the data structures and algorithms introduced in Ref. [12] are required.

We subsequently refer to the subgraph constructed in Ref. [12] as a *time step graph*, or TS graph. The output of analyzing this graph is the set of connected components corresponding to a set of vortices, which can be described by an ordered set of points or splines. We refer to the TS graph from the beginning and end of the time interval as the current TS graph, $TS_{current}$, and next TS graph, TS_{next}, respectively. The current and next TS graphs are connected by a *time interval graph*, or TI graph. A TI graph is the graph structure spanning a single time interval between two time steps, or only the nodes and edges fully inside one of the highlighted regions of Figs. 5 and 6.

TABLE I: Variables Used in Tracking Algorithm

$\psi_{current}$	Order parameter mesh data of current time step
ψ_{next}	Order parameter mesh data of next time step
$C_{current}$	Connected components of the current TS graph
C_{next}	Connected components of the next TS graph
$L_{current}$	Set of labels $\{l_{current}\}$ assigned to each connected component of $TS_{current}$
L_{next}	Set of labels $\{l_{next}\}$ assigned to each connected component of TS_{next}
L_{new}	Set of new labels $\{l_{new}\}$
B	Binary association matrix $n_{current} \times n_{current}$, where $n_{current}$ is the size of $L_{current}$
$F : l_{next} \to \{l_1, l_2, ...\}$	Mapping of each label $l_{next} \in L_{next}$ to labels $l_i \in L_{current} \cup L_{new}$

A TI graph has edges activated by both punctured time faces and space faces. Thus, an important step in constructing the TI graph is lifting the current TS graph into the TI graph. We observe that each node in a TS graph corresponds to an edge in the full dual graph to the 4D mesh, namely, the vertical edges spanning the highlighted regions in Figs. 5 and 6. Thus each node in the TS graph directly "activates" a single node in the following (or preceding) TI graph. Similarly each edge of the TS graph corresponds to one bundle in the TI graph and so "activates" a single edge in the TI graph. Although the meaning of the nodes and edges changes slightly when the TS graph is lifted to a TI graph, the graph structure is the same except for direction of the edges. Thus, the first step in constructing the TI graph is simply to make a copy of the current TS graph, while making each edge unidirectional. The rest of the nodes and edges of the TI graph are added by finding punctured time faces. Connected components are found in the TI graph by a queue-based flood-fill algorithm that also tracks when the flooding operation finds a differently labeled node. As a final step, the TI graph is compared with the lifted next TS graph.

Below we provide the algorithmic steps in detail for constructing the two TS graphs, their TI graph and interpreting events. The variables used below are defined in table I.

1. $TS_{current}$ and TS_{next} are constructed by performing contour integrals over all faces of the 3D meshes for $\psi_{current}$ and ψ_{next}. A connected component analysis is used to find the set of connected components $C_{current}$ and C_{next}. Each connected component in $C_{current}$ and C_{next} is given a unique label, $l_{current} \in L_{current}$ and $l_{next} \in L_{next}$

2. An empty TI graph is created and initialized by lifting the nodes and edges of $TS_{current}$ and labeling each lifted node $l_{current}$, where $l_{current} \in L_{current}$ is the label of its connected component in $C_{current}$

3. By using $\psi_{current}$ and ψ_{next}, contour integrals are performed around all time faces, one for each edge of the spatial mesh. For each punctured time face, corresponding edges and (unlabeled) nodes are added to TI graph.

4. A connected-component analysis of the TI graph is performed by using a flood-fill algorithm from the labeled nodes. If the graph traversal reaches a labeled node with a different label, the binary association matrix is updated. That is, if i, j are the labels of the two components, $i, j \in L_{current}$, set $B(i, j) = B(j, i) = 1$. Then, the binary association matrix B is updated so that it is fully associative. That is, if $B(i, j) = 1$ and $B(j, k) = 1$, then $B(i, k) = 1$.

5. Unlabeled connected components in the TI graph are births [type I in Fig.7(b)]. They are assigned a new label $l_{new} \in L_{new}$, and their nodes are labeled accordingly.

6. The mapping function F is now constructed by iterating over all nodes of the lifted TS_{next}, for a node labeled l_{next}.

   6.1. If the TI graph has an equivalent node with label l, then add l to the list l_{next} maps to. $F : l_{next} \rightarrow \{..., l\}$. If, for any $l' \neq l$, $B(l', l) = 1$, then also add l' to the list l_{next} maps to. $F : l_{next} \rightarrow \{..., l'\}$.

   6.2. Rarely (in 2D cases), the TI graph may not have an equivalent node. This is also a birth [type II in Fig. 7(b)] and should be handled analogously to (5), that is, assigned a new label $l_{new} \in L_{new}$ and added to the mapping $F : l_{next} \rightarrow \{..., l_{new}\}$.

The data structures F and B now contain all the information necessary to determine what events occurred from the current to the next time step.

1. *Continuity*: If, per F, for a given $l_{next} \in L_{next}$, $F : l_{next} \rightarrow \{l_{current}\}$, where $l_{current} \in L_{current}$ and no other $l \in L_{next}$ also maps to $l_{current}$, then l_{next} corresponds to a vortex that has continued.

2. *Births*: If, per F, for a given $l_{next} \in L_{next}$, $F : l_{next} \rightarrow \{..., l_{new}..., \}$, where $l_{new} \in L_{new}$, then l_{next} is a birth.

3. *Splits*: If, per F, more than one l_{next} maps to $l \in L_{current} \cup L_{new}$, then l has split.

4. *Merges*: If, per F, l_{next} maps to more than one $l \in L_{current} \cup L_{new}$, then l_{next} is a merge.

5. *Deaths*: If, per F, no $l_{next} \in L_{next}$ maps to $l_{current} \in L_{current}$, then vortex $l_{current}$ has died. If there is a $l' \neq l_{current}$ such that $B(l_{current}, l') = 1$ and l' also died, then a Merge-Death (Annihilation) has occurred.

This algorithm can be easily adapted so that it can be repeated over subsequent time steps by mapping each local label to a globally unique label and assigning new globally unique labels to vortices that are the result of births, merges, or splits.

In Figure 7, we illustrate the events of continuity [Fig. 7(a)], birth (type I and II) [Fig. 7(b)], death [Fig. 7(b)], birth-split [Fig. 7(c)], split [Fig. 7(c)], merge [Fig. 7(d)], merge-death [Fig. 7(e)], and merge-split [Fig. 7(f)], as they would appear using the two lifted TS graphs and the TI graph. We note that, in general, merges, merge-splits, and splits do not occur in 2D meshes with point vortices but do occur for 3D with line vortices where each vortex in a TS graph is composed of many nodes and edges.

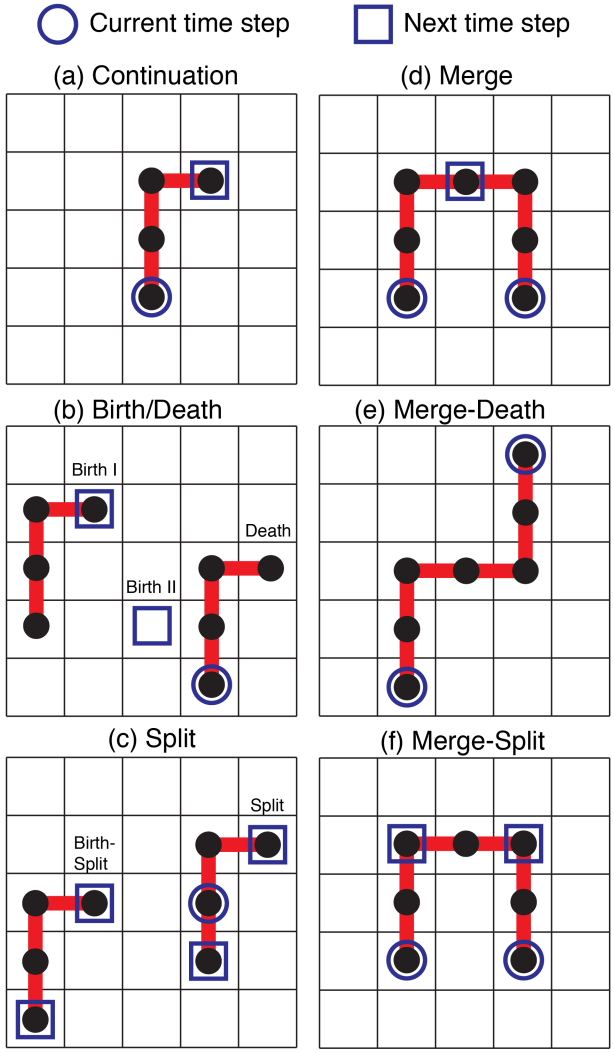

FIG. 7: (Color online) Events that can be detected in simulation, shown for a 2D mesh. The lifted current and next TS graphs (in 2D, a single nodes) are shown as a ○ and □, respectively. The TI graph is shown as a set of nodes (filled circles) and edges (red lines). Only the activated nodes and edges of the dual graph are shown.

A. Scaling and Timing

In Ref. [12], the scaling of the vortex extraction algorithm was characterized extensively with respect to increasing the mesh size and increasing the number of vortices present in the system. When extracting vortices from a single time step, the following conclusions were drawn. In a dilute vortex state, with a small, fixed number of vortices to find, the bulk of the algorithm time is performing contour integrals around space faces. In a dense vortex state, the bulk of the algorithm time is spent tracing the vortices in the graph, interpolating the points of the vortices on each face, and fitting curves to the vortices. Tracking of vortices between two time steps adds three significant computational steps, each directly analogous to calculations performed for vortex extraction. The first is the calculation of the contour integral around all time faces. Just as in [12], the number of calculations performed is proportional to the number of points in the mesh and will dominate the computation for a dilute vortex state where the vortices do not move much. The second and third additional computations are the construction of the TI graph and the flood-fill of the TI graph. Here, the number of calculations performed is proportional to the number of punctured time faces or the total number of nodes in TI graph. In a dense vortex state, or when the vortices have moved a large amount between the two time steps, this calculation will dominate the algorithm time. In comparison, detecting and recording events, which are performed on a much smaller data structure than the mesh and graph, are computationally negligible.

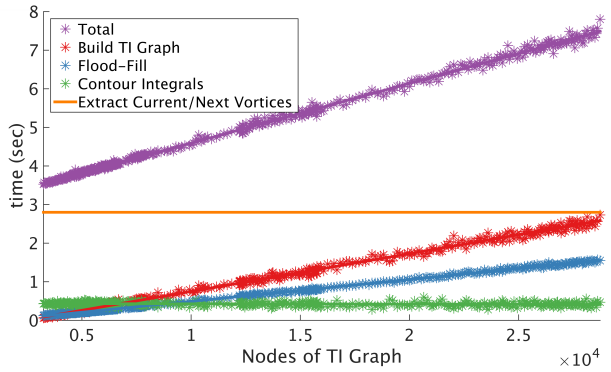

FIG. 8: (Color online) Scaling of the tracking algorithm shown with respect to increasing the size of the TI graph. A larger TI graph corresponds to faster moving or larger vortices.

In Figure 8, we show the computation time for the three major computational steps of tracking as a function of the number of nodes in the TI graph. This data was generated for a structured mesh of size 256x128x32. By choosing time steps progressively farther apart, the vortices present in the system moved farther and thus generate a progressively larger TI graph. Note that the minimum number of nodes of the TI graph is always greater than zero, since the graph consists of at least the same number of nodes as the first TS graph. Since the size of the mesh is fixed, the time required to calculate the contour integrals is also fixed. Similarly, the time to extract the vortices from the first and second time steps is fixed and is shown as a line. As the number of nodes of the TI graph increases, the time to construct the TI graph and time to flood-fill the TI graph increase linearly and eventually will dominate the calculation.

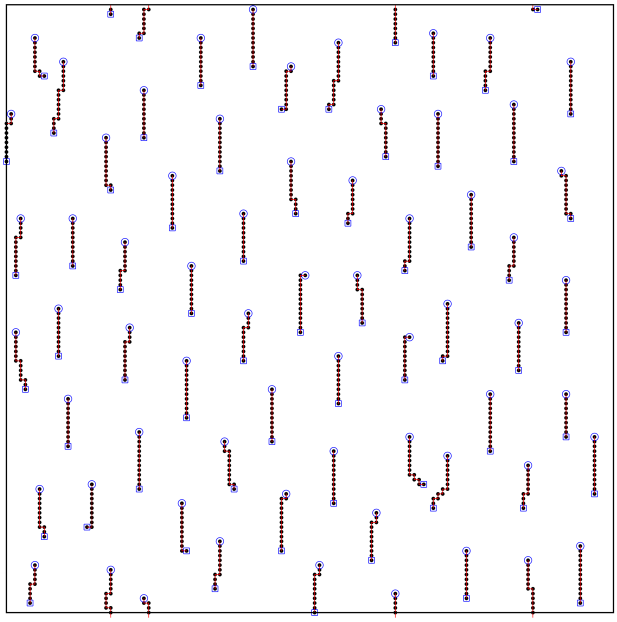

FIG. 9: (Color online) Two time steps of a set of 64 vortices in a two-dimensional simulation flowing downwards. The vortices in the first and second time step are circles and squares, respectively. Also shown are the black nodes of the TI graph that connects the vortices from the two time steps.

We note that if the vortices have moved a small amount between the two time steps, the computational cost of tracking a vortex between two time steps adds only a small amount of overhead. However, if the time interval between the two time steps is large enough that the vortices have moved a significant amount, then constructing and flood-filling the TI graph dominate the calculation. In general, using a large time interval can compromise the accuracy of the tracking because, when different vortices have moved through the same regions over a time interval and therefore punctured the same faces of the space-time mesh elements, tracking results may indicate merges and splits rather than continuity. In a system where vortices move at a roughly steady-state velocity, one can optimize the rate of tracking so as to continuously track vortices for the least computational effort.

We conclude that the primary cost of tracking vortices in simulation data is not the cost of tracking in addition to extracting the vortex but, rather, the frequency with which tracking calculations need to occur so as not to have too many events combine over an elapsed time-interval such that sequence of events and identity of each vortex becomes unclear. Optimizing the frequency of performing the tracking analysis is a problem-dependent assessment.

VI. EXAMPLES OF VORTEX TRACKING

Using data from a 2D simulation of point vortices and a 3D simulation of line vortices, we will demonstrate how the tracking algorithm can detect continuity and a merge-split.

A. Tracking vortices in 2D simulation

Figure 9 shows the tracking of a two-dimensional simulation of vortices. All data was generated by using the TDGL code described in Ref. [6]. Here, vortices are point defects in the ψ phase field. An x-directed superconducting current applies a Lorentz force on the vortices, causing them to flow downwards. The simulation has periodic boundaries, so vortices that exit the bottom of the simulation box appear at the top. No events other than continuity occur over the simulation, so the identity of each vortex is conserved. The 2D mesh is of size 128x128. Each mesh element is 0.5 coherence lengths (the unit of length in a TDGL simulation) on a side, and the two time steps are separated by 9.9 dimensionless time units, defined in Ref. [6]. This represents a coarsening of the time discretization, since in the simulation the TDGL equations were solved over 99 time steps spanning this time interval. The lifted vortices from the first time step and second time step are open circles and squares, respectively. The nodes of the TI graph are shown connected by red edges. Because the vortices in the simulation are single points, they trace a one-dimensional path in time and correspond to a connected component where each node has connectivity one or two. The point vortices have traveled sufficiently far over the time interval that, if given only their positions at the two time steps, it is difficult to determine which vortex maps to which vortex without employing assumptions about how the vortices were moving. In comparison, using the algorithm presented above, no assumptions about how the vortices moved are needed in order to map vortices from one time step to the next.

Additionally, we note that by indicating which edges, corresponding to punctured time faces, of the 2D spatial mesh were crossed, the connected graph associated with each vortex contains details about the trajectory of the vortex between the two time steps. Some vortices in Fig. 9, for example, did not travel in straight lines between the two time steps. By interpolating within a time face, as was done for space faces in Ref. [12], a trajectory can be reconstructed as a sequence of (x_i, y_i, t_i), where each t_i corresponds to the approximate time the vortex crossed an edge of the 2D mesh. This method was used to construct the smooth trajectories shown in Fig. 12 of Ref. [43].

B. Tracking vortices in 3D simulation

Figure 10 shows two vortices in a 3D TDGL simulation that recombine to form two new vortices by a merge-split. All data was generated by using the TDGL code described in Ref. [6]. Here vortices are 1D curves. In Figure 11, we show how the TI graph constructed for this time interval indicates this is a merge-split event. The 3D mesh is of size 256 x 128 x 32, although only a small portion of it is shown. Mesh elements are 0.5 coherence lengths on each side, and the two time steps are separated by 9.0 dimensionless time units or 90 time steps of the TDGL simulation spanning this time interval. The lifted vortices from the first time step and second time step are open circles and squares, respectively. The nodes of the TI graph are shown connected by edges. The vortices sweep out a connected fabric of nodes over the time interval. From the top image of Fig. 11, one can infer that Vortex 3 is composed of the right and left parts of Vortex 1 and Vortex 2, respectively, and Vortex 4 is composed of the left and right parts of Vortex 1 and Vortex 2, respectively. In other words, in this merge-split event, the two vortices swapped parts, rather than one vortex cutting, crossing the other vortex, and reforming. In the bottom of Fig. 11, the TI graph is shown from a side (and Vortex 1 and Vortex 2 are not shown) so that the connected TI graph can be seen more clearly and it is apparent that Vortex 3 and Vortex 4 do not intersect.

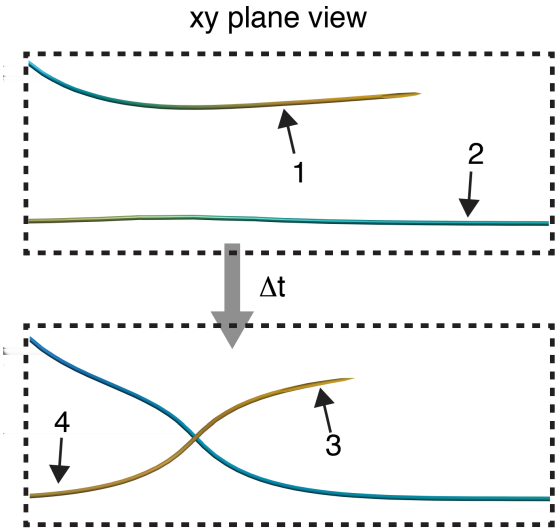

FIG. 10: (Color online) (Top) Merge-split that occurs between two vortices in a 3D mesh, viewed along the z-axis. Vortex 1 and 2 in the first time step become Vortex 3 and 4, in the second time step by cutting and recombining.

VII. CONCLUSION

In this paper we have presented a method that can track topological defect lines from a data set of complex scalars defined over a 4D space-time mesh at the scale of the discretization. In our application, the topological defects correspond to vortices in a TDGL simulation of a type II superconductor. Vortices are tracked by interpreting the set of topological defects as a connected subgraph of the dual graph of the 4D space-time mesh. Nodes and edges of this subgraph are constructed by performing integrals along closed paths on faces of the mesh. By analyzing how the graph structure changes from one time step to the next, events such as birth, death,

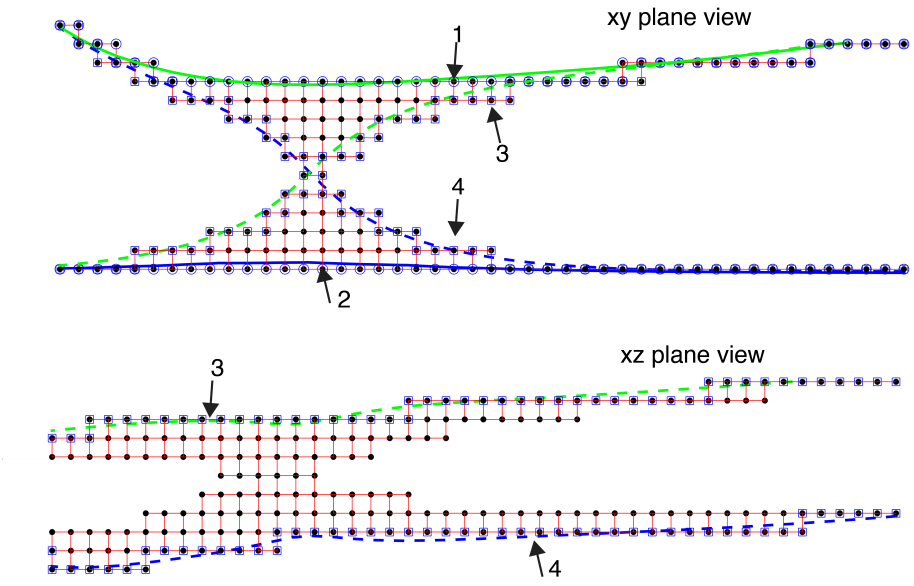

FIG. 11: (Color online) (Top) The merge-split of Figure 10 shown as a TI graph. Vortex 1 and Vortex 2 in the first time step are shown as solid green and sold blue lines, respectively. Vortex 3 and Vortex 4, in the second time step are shown as dashed green and dashed blue, respectively. The lifted nodes are shown as circles and squares for the first and second time steps, respectively. Also shown are the black nodes and red edges of the TI graph that connects the vortices from the two time steps. The 3D mesh is not shown. (Bottom) Only Vortex 3 and Vortex 4, the TI graph, and the lifted second time step nodes are shown, viewed along the y-axis. Vortex 3 and Vortex 4 are distinct because there is a gap between them along the z-axis.

continuity, merging, splitting, birth-splits, merge-deaths, and merge-splits can be detected. These events correspond to vortices spontaneously appearing or disappearing in a type-II superconductor and interacting with other vortices. While the implementation described here is for a regular structured mesh that is aligned along the Cartesian axes, we have also generalized this method to an unstructured mesh, where the edge-to-vertex dual of the subgraph is used instead [43].

Because vortices lack fixed features, length scale, movement patterns, or even conserved identities, standard methods of tracking objects can quickly fail under common circumstances. The tracking analysis presented here fails to track vortices and correctly detect events only if the time interval between time steps is too large. This tracking analysis permits vortex interactions to be understood at a finer detail than was previously possible and allows vortices to be tracked unambiguously over time. This analysis also supports creating a reduced representation of the narrative of the vortex dynamics. As TDGL simulations increase in size, in order to model experimentally relevant mesoscale superconducting phenomena, it will be important to be able to extract on the fly and visualize the dynamical narrative of how the vortices behave; otherwise the volume of simulation data will quickly overwhelm storage resources.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. C.L.P. was partially funded by the Office of the Director through the Named Postdoctoral Fellowship Program (Aneesur Rahman Postdoctoral Fellowship), Argonne National Laboratory. The work was also supported by the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Science.

Appendix A: Converting to edge-to-vertex dual

Because of the available data structures used in some mesh libraries, using the edge-to-vertex dual of the graph can be more computationally efficient for tracking vortices. This is the data structure used in Ref. [43]. With a simple procedure, a dual graph for a 4D mesh can be converted to its edge-to-vertex dual.

Concisely, in the dual graph described above, edges corresponds to a bundle of punctured faces. In the edge-to-vertex dual of this graph, each punctured face is a node; and if two punctured faces are shared in a bundle, then an edge connects their associated nodes. In Fig. 12, the edge-to-vertex dual of the graphs in Fig. 5 and Fig. 6 are generated.

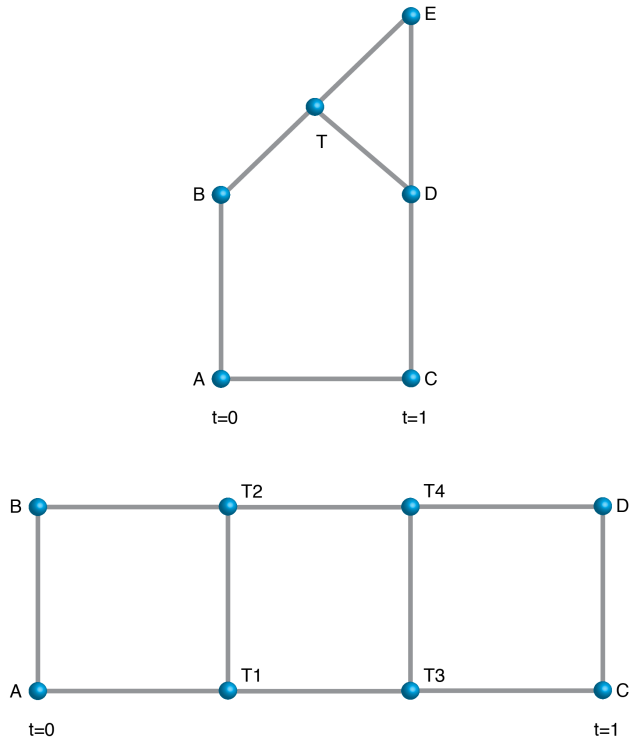

FIG. 12: (Color online) Edge-to-vertex dual of the graph of Fig. 5 (top) and Fig. 6 (bottom).

In the edge-to-vertex dual graph, edges between nodes that belong the same time step can be treated as directed edges, where the direction of the edge indicates the chirality of the vortex puncturing the two face. The direction of the edge is determined from the chirality of the punctured face (i.e., $n = \pm 1$) relative to the direction of the contour integral. We omit indicating the direction of the edges in Fig. 12 since edge directions are not used when tracking vortices.

Appendix B: Gauge-invariant vortex detection around space and time contours with a varying magnetic field and current

For a superconductor described by the order parameter $\psi = |\psi| e^{\imath\theta}$, the local vorticity is defined as

$$n \equiv -\frac{1}{2\pi} \oint_{\mathscr{C}} d\mathbf{l} \cdot \nabla\theta \,, \tag{B1}$$

along a closed contour $\mathscr{C}$ with $\mathscr{C} = \partial\mathscr{A}$ ($\mathscr{A}$ being the area enclosed by contour $\mathscr{C}$).

Whereas the magnitude of ψ is gauge-invariant, however, the phase of ψ is not. In Ref. [12], the above line integral was reformulated in a gauge-invariant manner as

$$n = -\frac{1}{2\pi} \left(\oint_{\mathscr{C}} d\mathbf{l} \cdot (\nabla\theta + K(t)\hat{x} - \mathbf{A}) + \int_{\mathscr{A}} \mathbf{B} \cdot d\mathbf{a} \right), \tag{B2}$$

where $\mathbf{A}$ is the magnetic vector potential, $K(t)$ is the time-varying time integral of the electric field in the x-direction, and $\mathbf{B}$ is the magnetic field.

The contour integral in Eq. (B2) can be exactly calculated over a set of connected segments $\{l_i\}$ forming a closed path, where θ is θ_{i-1} and θ_i at the endpoints of segment l_i, as long as $\tilde{\theta}$ does not change by more than π along any one segment. That is,

$$n \equiv -\frac{1}{2\pi} \left(\sum_{1}^{m} \Delta\tilde{\theta}_{i,i-1} + \int_{\mathscr{A}} \mathbf{B} \cdot d\mathbf{a} \right), \tag{B3}$$

where

$$\Delta\tilde{\theta}_{i,i-1} = \mathrm{mod}(\theta_i - \theta_{i-1} \tag{B4}$$

$$+ (K(t)\hat{x} - (\mathbf{A}_i - \mathbf{A}_{i+1})) \cdot l_i + \mathscr{Q}_{i,i-1} \tag{B5}$$

$$+ \pi, 2\pi) - \pi. \tag{B6}$$

In the expression (B5), $\mathbf{A_i}$ and $\mathbf{A_{i-1}}$ are the magnetic vector potential at the endpoints of l_i. For completeness, we also explicitly include the phase jump correction, $\mathscr{Q}_{i,i-1}$, for contour segments that cross a quasiperiodic boundary; that is, $\mathscr{Q}_{i,i-1} = 0$ unless l_i crosses a quasiperiodic boundary. We group the correction terms together as follows:

$$\Delta\tilde{\theta}_{i,i-1} = \mathrm{mod}(\theta_i - \theta_{i-1} + C_l(x,y,z,l_i,t) + \pi, 2\pi) - \pi. \tag{B7}$$

Since there is no gauge transformation or quasiperiodic boundary conditions on time edges, $C_l = 0$ on time edges, $l = \Delta t\, \hat{t}$, of the mesh. We note, however, that for a given spatial edge, C_l changes as a function of time if the magnetic field $\mathbf{B}$ or $K(t)$ changes. When calculating Eq. (B3) around a rectangular time face, that is, for a single spatial edge extended over a time interval, instead of calculating a magnetic flux through a face, a correction term is added to the summation that accounts for the change in C_l for the spatial segment l over the time interval:

$$n \equiv \frac{1}{2\pi} \left(-\sum_{1}^{m} \Delta\tilde{\theta}_{i,i-1} - \Delta C_{t+1,t} \right), \tag{B8}$$

where

$$\Delta C_{t+1,t} = \mathrm{mod}(C_l(x,y,z,l,t+1) - C_l(x,y,z,l,t) + \pi, 2\pi) - \pi. \tag{B9}$$

Again, if the magnetic field or K has not changed over the time interval, $\Delta C_{t+1,t} = 0$.

In the large λ-limit Ginzburg-Landau solver described in Ref. [6], the vector potential $\mathbf{A}$ was defined as a linear function in either the x or in the y direction. Using the formulation from Refs. [6] and [12], we can now explicitly write the correction term $C_l(x,y,z,l,t)$ of Eq. (B7) for contour integrals with segments on a Cartesian mesh for different magnetic field configurations. For simplicity of notation we will express $C_l(x,y,z,l,t)$ as $C_l(i,j,k,l,t)$, where $x = h_x i$, $y = h_y j$, and $z = h_z k$, where h_x, h_y, and h_z are the lengths of the sides of a mesh element.

For an **xz** magnetic field, $\mathbf{B} = [B_x, 0, B_z]$

$$\begin{aligned} l = h_x\hat{x}&: \\ &C_l(i,j,k,l,t) = B_z(t)\bar{y}(j)h_x + K(t)h_x \\ l = h_y\hat{y}&: \\ &C_l(i,j,k,l,t) = \mathscr{Q}(i,j,k,l,t) \\ l = h_z\hat{z}&: \\ &C_l(i,j,k,l,t) = -B_x(t)\bar{y}(j)h_z, \end{aligned} \tag{B10}$$

where

$$\begin{aligned} &\mathscr{Q}(i,j,k,l = h_y\hat{y},t) = \\ &(-L_yB_z(t)h_xi + L_yB_x(t)h_zk)\,\Theta((j+1)h_y - L_y) \end{aligned} \tag{B11}$$

and $\bar{y}(j) = h_y(j - \frac{n_y}{2})$. The Heaviside function $\Theta(x)$ is 1 if $x \geq 0$, and 0 otherwise. The purpose of Θ is to apply the quasiperiodic phase jump correction only if the path segment l crosses a quasiperiodic boundary.

Similarly, for an **yz** magnetic field, $\mathbf{B} = [0, B_y, B_z]$

$$\begin{aligned} l = h_x\hat{x}&: \\ &C_l(i,j,k,l,t) = K(t)h_x + \mathscr{Q}(i,j,k,l,t) \\ l = h_y\hat{y}&: \\ &C_l(i,j,k,l,t) = -B_z(t)\bar{x}(i)h_y \\ l = h_z\hat{z}&: \\ &C_l(i,j,k,l,t) = B_y(t)\bar{x}(i)h_z, \end{aligned} \tag{B12}$$

where

$$\begin{aligned} &\mathscr{Q}(i,j,k,l = h_y\hat{y},t) = \\ &(L_xB_z(t)h_yj - L_xB_y(t)h_zk)\,\Theta((i+1)h_x - L_x) \end{aligned} \tag{B13}$$

and $\bar{x}(i) = h_x(i - \frac{n_x}{2})$.

[1] J. Koplik and H. Levine, Phys. Rev. Lett. **71**, 1375 (1993).
[2] D. Samuels, C. Barenghi, and R. Ricca, Journal of Low Temperature Physics **110**, 509 (1998).
[3] J. Leach, M. R. Dennis, J. Courtial, and M. J. Padgett, Nature **432**, 165 (2004).
[4] D. Kleckner and W. Irvine, Nature Physics **9**, 253 (2013).
[5] M. V. Berry and M. R. Dennis, Journal of Physics A: Mathematical and Theoretical **40**, 65 (2007).
[6] I. Sadovskyy, A. Koshelev, C. Phillips, D. Karpeyev, and A. Glatz, Journal of Computational Physics **294**, 639 (2015).
[7] A. Glatz, H. L. L. Roberts, I. S. Aranson, and K. Levin, Phys. Rev. B **84**, 180501 (2011).
[8] V. Vlasko-Vlasov, A. Koshelev, A. Glatz, C. Phillips, U. Welp, and W. Kwok, Phys. Rev. B **91**, 014516 (2015).
[9] M. Bou-Diab, M. J. W. Dodgson, and G. Blatter, Phys. Rev. Lett. **86**, 5132 (2001).
[10] V. K. Vlasko-Vlasov, A. Glatz, A. E. Koshelev, U. Welp, and W. K. Kwok, Phys. Rev. B **91**, 224505 (2015).
[11] A. Koshelev, I. Sadovskyy, C. Phillips, and A. Glatz, arXiv:1509.04212.
[12] C. L. Phillips, T. Peterka, D. Karpeyev, and A. Glatz, Phys. Rev. E **91**, 023311 (2015).
[13] I. S. Aranson, A. R. Bishop, I. Daruka, and V. M. Vinokur, Phys. Rev. Lett. **80**, 1770 (1998).
[14] M. B. Hindmarsh and T. W. B. Kibble, Reports on Progress in Physics **58**, 477 (1995).
[15] I. S. Aranson and L. Kramer, Rev. Mod. Phys. **74**, 99 (2002).
[16] E. Hamm, S. Rica, and A. Vierheilig, in *Instabilities and Nonequilibrium Structures VI*, Nonlinear Phenomena and Complex Systems, Vol. 5, edited by E. Tirapegui, J. Martnez, and R. Tiemann (Springer, 2000) pp. 207–217.
[17] S. Madruga, H. Riecke, and W. Pesch, Phys. Rev. Lett. **96**, 074501 (2006).
[18] S. Ryu, S. Doniach, G. Deutscher, and A. Kapitulnik, Phys. Rev. Lett. **68**, 710 (1992).
[19] W. R. Magro and D. M. Ceperley, Phys. Rev. B **48**, 411 (1993).
[20] H. Nordborg and G. Blatter, Phys. Rev. Lett. **79**, 1925 (1997).
[21] H. Nordborg and G. Blatter, Phys. Rev. B **58**, 14556 (1998).
[22] A. van Otterlo, R. T. Scalettar, and G. T. Zimányi, Phys. Rev. Lett. **81**, 1497 (1998).
[23] A. E. Koshelev and A. B. Kolton, Phys. Rev. B **84**, 104528 (2011).
[24] D. Ertas and M. Kardar, Phys. Rev. B **53**, 3520 (1996).
[25] T. Chen and S. Teitel, Phys. Rev. B **55**, 11766 (1997).
[26] A. E. Koshelev, Phys. Rev. B **56**, 11201 (1997).
[27] A. K. Nguyen, A. Sudbø, and R. E. Hetzel, Phys. Rev. Lett. **77**, 1592 (1996).
[28] A. K. Nguyen and A. Sudbø, Phys. Rev. B **57**, 3123 (1998).
[29] A. K. Nguyen and A. Sudbø, Phys. Rev. B **58**, 2802 (1998).
[30] Y. Nonomura, X. Hu, and M. Tachiki, Phys. Rev. B **59**, R11657 (1999).
[31] A. K. Nguyen and A. Sudbø, Phys. Rev. B **60**, 15307 (1999).
[32] G. W. Crabtree, D. O. Gunter, H. G. Kaper, A. E. Koshelev, G. K. Leaf, and V. M. Vinokur, Phys. Rev. B **61**, 1446 (2000).
[33] Y. Nonomura and X. Hu, Phys. Rev. Lett. **86**, 5140 (2001).
[34] E. Coskun and M. K. Kwong, Nonlinearity **10**, 579 (1997).
[35] S. Kim, J. Burkardt, M. Gunzburger, J. Peterson, and C.-R. Hu, Phys. Rev. B **76**, 024509 (2007).
[36] X. H. Chao, B. Y. Zhu, A. V. Silhanek, and V. V. Moshchalkov, Phys. Rev. B **80**, 054506 (2009).
[37] Q. Du, Journal of Mathematical Physics **46**, 095109 (2005).
[38] F. Reinders, F. H. Post, and H. J. Spoelder, The Visual Computer **17**, 55 (2001).
[39] M. Jiang, R. Machiraju, and D. Thompson, in *The Visualization Handbook* (Academic Press, 2005) pp. 295–309.
[40] J. Jeong and F. Hussain, Journal of Fluid Mechanics **285**, 69 (1995).
[41] R. Peikert and M. Roth, in *Proc. of IEEE Visualization '99* (1999) pp. 263–270.
[42] H. Theisel and H.-P. Seidel, in *Proceedings of the Symposium on Data Visualisation 2003*, VISSYM '03 (Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003) pp. 141–148.
[43] H. Guo, C. Phillips, T. Peterka, D. Karpeyev, and A. Glatz, IEEE Transactions on Visualization and Computer Graphics **22**, 827 (2016).
[44] A. Kanda, B. J. Baelus, F. M. Peeters, K. Kadowaki, and Y. Ootuka, Phys. Rev. Lett. **93**, 257002 (2004).