Using PVsolve to Analyze and Locate Positions of Parallel Vectors

Allen Van Gelder, *Member*, *IEEE Computer Society*, and Alex Pang, *Senior Member*, *IEEE*

Abstract—A new method for finding the locus of parallel vectors is presented, called PVsolve. A parallel-vector operator has been proposed as a visualization primitive, as several features can be expressed as the locus of points where two vector fields are parallel. Several applications of the idea have been reported, so accurate and efficient location of such points is an important problem. Previously published methods derive a tangent direction under the assumption that the two vector fields are parallel at the current point in space, then extend in that direction to a new point. PVsolve includes additional terms to allow for the fact that the two vector fields may not be parallel at the current point, and uses a root-finding approach. Mathematical analysis sheds new light on the feature flow field technique (FFF) as well. The root-finding property allows PVsolve to use larger step sizes for tracing parallel-vector curves, compared to previous methods, and does not rely on sophisticated differential equation techniques for accuracy. Experiments are reported on fluid flow simulations, comparing FFF and PVsolve.

Index Terms—Parallel vectors, feature flow field, vortex core, flow visualization, PVsolve, adjugate matrix, Newton-Raphson root finding, dimensionless projection vector.

1 Introduction

In certain scientific visualization applications, especially those involving fluid flows, the question arises whether vectors in two different vector fields are parallel. In a seminal paper, Peikert and Roth [13] proposed a parallel-vector operation as a visualization primitive. They observed that several concepts of "vortex core" can be formulated as the set of points where the vectors drawn from two 3D fields are parallel, and extremal curves on surfaces can be similarly characterized. For two smoothly varying 3D vector fields, say $\mathbf{v}$ and $\mathbf{w}$, Roth observed that the set of points where $\mathbf{v} \| \mathbf{w}$, i.e., $\mathbf{v}$ is parallel to $\mathbf{w}$, normally forms some number of smooth curves [14]. Parallel-vector curves have been used in several recent applications. For example, Garth et al. [3] used this, among numerous other methods, to investigate the swirling and tumbling motion inside a diesel engine.

In their paper in 1999, Peikert and Roth identified four schemes for finding the parallel vectors in a pair of vector fields, as reviewed in Section 2. The fourth scheme, curve tracing, has seen recent activity and has been found to offer better numerical algorithms compared to the other approaches. Theisel and Seidel [20] reformulated the parallel-vector problem as streamline tracing in a *feature flow field* (FFF). For purposes of this paper, a "feature" is a point where the two vector fields are parallel, and a feature curve or feature surface is a connected set of such points. Theisel et al. [19] and Weinkauf et al. [23] extended the FFF idea, proposing new formulations, with vortex core lines as one of the target applications. Independently, Sukharev et al. [18] proposed an alternative formulation based on an "analytical tangent," which facilitates tracing of the feature curves.

This paper introduces a new methodology, called `PVsolve`, that adopts a somewhat different view of the tracing problem. Whereas the other published tracing-oriented methods view the problem as integrating an ordinary differential equation, we view it as root finding. The principal difference in approaches can be succinctly summarized in nontechnical language, as follows:

- FFF begins at a point where the cross product of the two vector fields is zero and ideally progresses to other points where the cross product is zero. If it gets a little off course, it tries to pursue a path on which the cross product *retains its current value*.
- `PVsolve` begins at a point where the dimensionless projection vector is zero and ideally progresses to other points where the dimensionless projection vector is zero. (The dimensionless projection vector results from projecting one normalized vector onto a plane perpendicular to the other; thus, its magnitude is the sine of the angle between the two vectors.) If it gets a little off course, it tries to move to a point at which the dimensionless projection vector *returns to zero*.

The details are presented in Section 3, but we note here that at points where the two vectors are parallel, both the cross product and the dimensionless projection vector are zero. At other points, they stand in a known relation to each other, but are not the same.

Although the FFF method has been found to be fairly robust, under unfavorable circumstances error can accumulate. Using fourth-order Runge-Kutta is not a panacea, as shown by Nielson and Jung [12] in another context. Essentially, each step introduces some tiny numerical error, moving to a point where the cross product is slightly changed. FFF tries to maintain that error at its current value

• *The authors are with the Computer Science Department, University of California, Santa Cruz, Santa Cruz, CA 95064.*

Manuscript received 9 Sept. 2008; accepted 9 Dec. 2008; published online 5 Jan. 2009.
Recommended for acceptance by M. Chen, C. Hansen, and K.-L. Ma.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2008-09-0145.
Digital Object Identifier no. 10.1109/TVCG.2009.11.

by keeping the cross product constant [20]. As shown in Section 5, maintaining the cross product at its current value, even though that value is very small, sometimes forces the trace to depart from the correct curve. In many cases, the method is successful because the trace approaches a fiducial point (i.e., a point known to be correct by other methods) before the error has grown significantly, and the error is reset to zero by continuing from the fiducial point. PVsolve also encounters numerical errors, but because of its root-finding orientation, it continuously reduces them.

In terms of mathematical techniques, the previously published methods consider homogeneous sets of linear equations, whereas PVsolve considers more general nonhomogeneous sets of linear equations, and the analysis is consequently more complex. However, by working at the abstract level of vectors and matrices, the PVsolve framework applies to all combinations of dimensions. That is, the two fields of nD vectors are defined over an mD euclidean space, and the main result applies to general combinations of n and m. The main cases of interest are $n = 3$ and $m = 3$ (3D space, fixed time) or $m = 4$ (time is the fourth "space" dimension).

The rest of the paper is organized as follows: Section 2 reviews the basic extraction and tracing strategy and also describes the differences and similarities among the different tracing approaches. Section 3 develops the mathematical basis for PVsolve. Section 4 illustrates the predictor-corrector nature of the method on an analytical example. Section 5 shows how FFF can occasionally go astray. Section 6 presents experimental comparison of PVsolve with FFF, using Runge-Kutta 4/5 as the integration method in FFF. Results on three fluid-flow simulation data sets are reported. Section 7 summarizes the paper and points to some future work.

2 Related Work

The four works most closely related to this paper are Banks and Singer [1], [2], Peikert and Roth [13], [14], Theisel et al. [19], and Sukharev et al. [18]. We discuss these variations in turn. We say that the locus of points where two vector fields are parallel are *feature curves* and points on such curves are *feature points*.

Banks and Singer's work is used in a number of vortex core tracing papers, including that of Peikert and Roth, and is therefore briefly summarized here. Their work involves two steps: an extraction (or seed point location phase) and a tracing phase using a predictor-corrector strategy. In the extraction phase, candidate seed points are found where grid points are found to have low pressure and a large magnitude of vorticity, i.e., those points meeting these two threshold values.

Once these seed points are found, each one is then used to trace a vortex core line. The prediction stage simply advances the seed point in both forward and backward directions along the vorticity vector. That new position is then corrected to a minimum-pressure point on the plane perpendicular to the vorticity vector at the predicted location. The search for the minimum pressure point is carried out via steepest gradient descent. They note that while vorticity or the pressure gradient fields are individually unreliable at capturing vortex core lines, the combination of these two is quite robust, even in cases where the flow transitions from laminar to turbulent flows. On the other hand, there is no guarantee that the gradient descent method will converge to the vortex core on the plane. In a related paper, Jeong and Hussain [8] combined the predictor-corrector approach with the λ_2 method to improve the efficiency in tracking vortex cores. A key modification was to replace the pressure field by the λ_2 field during the correction stage [16]. In addition, the gradient descent was replaced by a direct search approach [7].

Peikert and Roth's contribution is primarily to define the parallel-vector operator and demonstrate its expressiveness for capturing different interesting physical properties. One application is identification of vortex core lines. They describe how points on such vortex core lines can be specified in terms of certain vectors being parallel. While they identified four methods for constructing the locus of parallel vectors, Roth's dissertation provided only the heuristic-based approach of connecting extracted feature points. Briefly, they have a two-stage algorithm in which solution points (where vectors are parallel) are found in the first stage and connected in the second stage. Each data set is treated as a discrete set of 3D cells, and Newton-Raphson root finding is used to find solution points on each of the 2D faces of the cell. Simple connection rules are used in the second stage to connect points in faces of the same cell. If a cell contains only two solution points, they are simply connected. If a cell has four solution points, then they choose the two connections with the maximal distances from each other out of the three possible pairs of connections. Cells with six solution points are not handled, and cells with an odd number of solution points are subdivided until there are 0, 2, or 4 solution points in a cell. While this algorithm may seem simplistic, it was considered state of the art at that time. However, Sukharev et al. [18] showed cases in which the Peikert-Roth connection heuristics led to incorrect topology of the solution curve. Another limitation of the extract-then-connect approach is the need to extract all the solution points. Therefore, in practice, Peikert and Roth concluded that the preferred approach for finding parallel vectors is a modification of the extraction and tracing strategy described by Banks and Singer. In the tracing phase, an integration step is taken along $\mathbf{v}$ (or $\mathbf{w}$ since they are parallel at the seed point). Each step must immediately be followed by a correction step to make sure that the new point is still on the feature curve. For this, they propose a correction step on $\mathbf{N}$, the plane perpendicular to $\mathbf{v}$ (or $\mathbf{w}$ if that was the vector used in tracing). The correction step will seek to minimize the projection of $\mathbf{w}$ (or $\mathbf{v}$) onto $\mathbf{N}$.

The choice of direction $\mathbf{v}$ was heuristic since the tangent to the feature curve is different from the direction of $\mathbf{v}$, in general. Two subsequent papers derived the tangent direction analytically and we review these next.

Theisel et al. [19] also use an extraction and tracing strategy, but formulated differently. They note that parallel vectors can be formulated as an "FFF," as introduced by Theisel and Seidel [20] primarily for time-varying flow fields. Feature points correspond to places in space-time where vectors are parallel, and connected sets of points form curves in 3D or surfaces in 4D. This procedure is generalized in Weinkauf's dissertation [22].

Using the notation $\mathbf{s} = \mathbf{v} \times \mathbf{w}$ and $\mathbf{C} = \nabla(\mathbf{s})$, we now summarize the 3D fixed-time construction of Theisel et al. [19]. They construct a vector field $\mathbf{f}$ over 3D euclidean space

such that at a seed point $\mathbf{x}_0$ with $\mathbf{s}(\mathbf{x}_0) = \mathbf{0}$, a streamline of $\mathbf{f}$ through that point also satisfies $\mathbf{s}(\mathbf{x}) = \mathbf{0}$. Such an FFF $\mathbf{f}$ can be constructed by

$$\mathbf{f} = \begin{bmatrix} \det[\mathbf{C}_{*,2}, \mathbf{C}_{*,3}, \mathbf{a}] \\ \det[\mathbf{C}_{*,3}, \mathbf{C}_{*,1}, \mathbf{a}] \\ \det[\mathbf{C}_{*,1}, \mathbf{C}_{*,2}, \mathbf{a}] \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{*,2} \times \mathbf{C}_{*,3} \cdot \mathbf{a} \\ \mathbf{C}_{*,3} \times \mathbf{C}_{*,1} \cdot \mathbf{a} \\ \mathbf{C}_{*,1} \times \mathbf{C}_{*,2} \cdot \mathbf{a} \end{bmatrix}, \quad (1)$$

where $\mathbf{C}_{*,i}$ is the ith column of $\mathbf{C}$ and represents the first-order partial of the cross product $\mathbf{s}$ with respect to i. Also,

$$\mathbf{a} = \begin{cases} \mathbf{w}, & \text{if } |\mathbf{w}| \geq |\mathbf{v}| \\ \mathbf{v}, & \text{otherwise.} \end{cases} \quad (2)$$

In essence, they formulate the tangent direction of the feature curve, which allows them to integrate the feature curve starting from an extracted seed point. (Only the direction of $\mathbf{f}$ is significant, so it is normalized before use.) Note that some of the notations were changed from their original paper to be consistent with the notations used in this paper.

Sukharev et al. formulate an "analytical tangent" to trace the feature curve from a seed point $\mathbf{x}_0$ where the vectors are parallel. Such a tangent $\mathbf{t}$ can be obtained by solving for $\mathbf{t}$ in

$$s = \pm \frac{|\mathbf{v}|}{|\mathbf{w}|}, \quad (3)$$

$$\mathbf{A} = \nabla(\mathbf{v}) - s\nabla(\mathbf{w}), \quad (4)$$

$$\mathbf{A}\mathbf{t} = \mathbf{v}, \quad (5)$$

where the sign of s depends on whether the two vectors point in the same direction or not. Rather than compute $\mathbf{A}^{-1}\mathbf{v}$, the authors omit the division by $\det(\mathbf{A})$ since they are interested in the direction of the tangent line and not its magnitude (see Section 3.4). This also avoids singularity points. Thus, the components of the analytical tangent are

$$\mathbf{t} = \begin{bmatrix} \det[\mathbf{v}, \mathbf{A}_{*,2}, \mathbf{A}_{*,3}] \\ \det[\mathbf{A}_{*,1}, \mathbf{v}, \mathbf{A}_{*,3}] \\ \det[\mathbf{A}_{*,1}, \mathbf{A}_{*,2}, \mathbf{v}] \end{bmatrix}, \quad (6)$$

where $\mathbf{A}_{*,i}$ is the ith column of $\mathbf{A}$.

While developed independently about the same time and derived from different viewpoints, the formulations of the tangent directions given in (1) and (6) have strong similarities. In Section 3, we show that, even though the equations have some differences, both $\mathbf{f}$ and $\mathbf{t}$ point in the same direction at points where the underlying vector fields are parallel. Both formulations are valid when the trace is started from a seed point or a traced point that is *on the feature curve*.

3 Parallel Vectors within Vector Fields

This section develops the mathematical basis for our parallel-vector solving method, which we call `PVsolve`. It provides a uniform framework for solving several problems involving parallel vectors. We say that the locus of points where two vector fields are parallel is a *feature point set*, in general; it is called a *feature curve* or *feature line* when the point set is 1D; it is called a *feature surface* when the point set is 2D.

We then show how it is related to other recent tracing methods, which seek to follow feature curves in 3D. In particular, we show that they are essentially equivalent for the case when the trace is from a point on the feature curve, but our new formulation also handles the case where the trace has to continue from a point that is slightly off the feature curve. That is, within certain bounds, a correction step in the tracing phase allows our algorithm to home back to the feature curve.

Our development is fairly general in that we consider fields of nD vectors defined over an mD euclidean space. This generality is useful because there tend to be several values of m that are of interest for each n, and not assuming $m = n$ ensures that we do not rely on special cases, such as a 3D cross product. Our primary interest is $n = 3$, but for this n, we are interested in three values of m:

1. When $m = 2$, the problem is to find points in a planar cell face.
2. When $m = 3$, the problem is to find curves in 3D.
3. When $m = 4$, the fourth "spatial" dimension is time, and the problem is to find a 2D surface in 4D.

Although we are careful to specify m and n in the definitions, the reader usually does not need to worry about these values because vector and matrix notation is used. Vectors are column vectors in our notation, unless specified otherwise, and superscript T denotes transpose for vectors and matrices. Usually $\mathbf{x}$ denotes a point in mD euclidean space, while $\mathbf{v}$ and $\mathbf{w}$ are the nD vector fields of interest. The inner product of $\mathbf{v}$ and $\mathbf{w}$ is $\mathbf{v}^T\mathbf{w}$. The gradient operator, denoted by ∇, is a row vector operator

$$\nabla(\mathbf{v}) = \left[\frac{\partial \mathbf{v}}{\partial \mathbf{x}_1}, \frac{\partial \mathbf{v}}{\partial \mathbf{x}_2}, \ldots, \frac{\partial \mathbf{v}}{\partial \mathbf{x}_m}\right]. \quad (7)$$

So the gradient of a scalar is a $1 \times m$ row vector and the gradient of an nD (column) vector $\mathbf{v}$ is an $n \times m$ matrix, which we denote by $\nabla(\mathbf{v})$, as in several earlier papers on parallel vectors. This matrix is also known as the Jacobian matrix, sometimes written as $\mathbf{J}_\mathbf{v}(\mathbf{x})$, but our ∇ notation follows earlier papers on parallel vectors. We use $\mathbf{A}_{i,*}$ and $\mathbf{A}_{*,j}$ to denote the ith row and jth column of matrix $\mathbf{A}$, respectively. Golub and Van Loan [4] provide an excellent review of matrix and vector operations with attention to computational issues.

The matrix manipulations in the subsequent subsections are mostly applications of the identities reviewed in (8)-(15), where α denotes a scalar, $\mathbf{v}$ and $\mathbf{w}$ denote vectors, and $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ denote matrices (including vectors, as matrices of one column or one row)

$$\mathbf{v}^T\mathbf{w} = \mathbf{w}^T\mathbf{v}, \quad (8)$$

$$\alpha\mathbf{A} = \mathbf{A}\alpha, \quad (9)$$

$$(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T\mathbf{A}^T, \quad (10)$$

$$(\mathbf{A}\mathbf{B})\mathbf{C} = \mathbf{A}(\mathbf{B}\mathbf{C}), \quad (11)$$

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = (\mathbf{A}\mathbf{B}) + (\mathbf{A}\mathbf{C}). \quad (12)$$

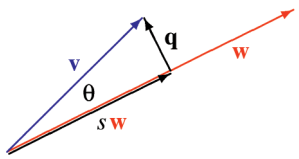


Fig. 1. Schematic showing relationship between $\mathbf{v}$, $\mathbf{q}$, θ, and $s\mathbf{w}$.

Vector forms of the product and quotient rules for differentiation are

$$\nabla(\mathbf{v}^T\mathbf{w}) = \mathbf{v}^T\nabla(\mathbf{w}) + \mathbf{w}^T\nabla(\mathbf{v}), \tag{13}$$

$$\nabla(\alpha\mathbf{w}) = \alpha\nabla(\mathbf{w}) + \mathbf{w}\nabla(\alpha), \tag{14}$$

$$\nabla\left(\frac{\mathbf{w}}{\alpha}\right) = \frac{\nabla(\mathbf{w})}{\alpha} - \frac{\mathbf{w}\nabla(\alpha)}{\alpha^2}. \tag{15}$$

Also, it is important not to confuse $\mathbf{v}\mathbf{w}^T$, which is a matrix and is called the outer product, with $\mathbf{v}^T\mathbf{w}$, which is the scalar inner product. Note that $\mathbf{w}\nabla(\alpha)$ is an outer product in (14) and (15).

At a high level, the root finding technique we use is the standard Newton-Raphson method, which can be applied to any vector function $\mathbf{g}$ that measures how far two vector fields are from being parallel, and is $\mathbf{0}$ when they *are* parallel. Considering $\mathbf{v}$ and $\mathbf{w}$ to be the two vector fields of interest, suppose we know their values and their gradients $\nabla(\mathbf{v})$ and $\nabla(\mathbf{w})$ at a point $\mathbf{x}_0$. We use subscript 0 for quantities evaluated at $\mathbf{x}_0$. We can approximate the values in the neighborhood with first-order Taylor series

$$\mathbf{g}(\mathbf{x}_0 + \delta\mathbf{x}) = \mathbf{g}(\mathbf{x}_0) + \nabla(\mathbf{g})_0\delta\mathbf{x}. \tag{16}$$

In other words, if

$$\mathbf{g}(\mathbf{x}_0) + \nabla(\mathbf{g})_0\delta\mathbf{x} = \mathbf{0}, \tag{17}$$

then $x_0 + \delta\mathbf{x}$ is a point where $\mathbf{v}$ and $\mathbf{w}$ are expected to be parallel (based on first-order terms). In the course of the paper, we shall instantiate $\mathbf{g}$ to several functions. To use (17), it is necessary to have workable expressions for $\nabla(\mathbf{g})$. This is the objective of Section 3.1.

3.1 Mathematical Development

Let $\mathbf{v}$ and $\mathbf{w}$ be smoothly varying vector fields, and suppose $|\mathbf{w}| \neq \mathbf{0}$ locally (otherwise interchange their roles in the following). Throughout this paper, θ denotes the absolute angle between $\mathbf{v}$ and $\mathbf{w}$, which is between $0°$ and $180°$. Decompose $\mathbf{v}$ into a component parallel to $\mathbf{w}$, denoted by $s\mathbf{w}$, and a component $\mathbf{q}$ that is orthogonal to $\mathbf{w}$ (see Fig. 1). Then, by the well-known Gram-Schmidt procedure, s and $\mathbf{q}$ can be expressed as

$$s = \frac{\mathbf{v}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}}, \tag{18}$$

$$\mathbf{q} = \mathbf{v} - s\mathbf{w} = \mathbf{v} - \mathbf{w}\left(\frac{\mathbf{w}^T\mathbf{v}}{\mathbf{w}^T\mathbf{w}}\right) = \mathbf{P}_\mathbf{w}\mathbf{v}, \tag{19}$$

$$\mathbf{P}_\mathbf{w} = \left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\mathbf{w}^T\mathbf{w}}\right). \tag{20}$$

The matrix $\mathbf{P}_\mathbf{w}$ is the well-known *projection operator* for an orthogonal projection of any nD vector onto a hyperplane orthogonal to $\mathbf{w}$.

Clearly, $\mathbf{q} = \mathbf{0}$ if and only if $\mathbf{v}$ and $\mathbf{w}$ are parallel. Toward the goal of instantiating $\mathbf{g}$ in (17) either to $\mathbf{q}$ or to an expression involving $\mathbf{q}$, we first derive an expression for $\nabla(\mathbf{q})$. In the following lemma, the term with $\mathbf{q}^T$ does not appear in earlier formulas related to tracing parallel vectors. This term makes the expression for $\nabla(\mathbf{q})$ correct for all $\mathbf{x}$, not just points where $\mathbf{q} = \mathbf{0}$.

The proof is presented in detail, so interested readers can confirm that all the manipulations are correct for general n and m. In particular, there are no special cases that require 3D (e.g., cross products) or that assume square matrices (e.g., matrix inverses). Readers may skip the proof without loss of continuity.

Lemma 3.1. *Using the terminology defined in (18)-(20),*

$$\nabla(\mathbf{q}) = \mathbf{P}_\mathbf{w}\big(\nabla(\mathbf{v}) - s\nabla(\mathbf{w})\big) - \frac{\mathbf{w}\mathbf{q}^T}{(\mathbf{w}^T\mathbf{w})}\nabla(\mathbf{w}). \tag{21}$$

Proof. Applying the definitions in (19) and (18), as well as (14)

$$\nabla(\mathbf{q}) = \nabla(\mathbf{v}) - \nabla(s\mathbf{w}) = \nabla(\mathbf{v}) - s\nabla(\mathbf{w}) - \mathbf{w}\nabla(s). \tag{22}$$

Next comes a careful derivation to rewrite $\nabla(s)$ into a usable form

$$\nabla(s) = \nabla\left(\frac{\mathbf{v}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}}\right) = \frac{\nabla\big(\mathbf{v}^T\mathbf{w}\big)}{\mathbf{w}^T\mathbf{w}} - \frac{(\mathbf{v}^T\mathbf{w})\nabla\big(\mathbf{w}^T\mathbf{w}\big)}{\big(\mathbf{w}^T\mathbf{w}\big)^2} \tag{23}$$

$$= \frac{\nabla\big(\mathbf{v}^T\mathbf{w}\big)}{\mathbf{w}^T\mathbf{w}} - \frac{s}{\mathbf{w}^T\mathbf{w}}\nabla\big(\mathbf{w}^T\mathbf{w}\big) \tag{24}$$

$$= \frac{\mathbf{v}^T\nabla(\mathbf{w}) + \mathbf{w}^T\nabla(\mathbf{v}) - s\big(\mathbf{w}^T\nabla(\mathbf{w}) + \mathbf{w}^T\nabla(\mathbf{w})\big)}{\mathbf{w}^T\mathbf{w}} \tag{25}$$

$$= \frac{1}{\mathbf{w}^T\mathbf{w}}\big(\mathbf{q}^T\nabla(\mathbf{w}) + \mathbf{w}^T(\nabla(\mathbf{v}) - s\nabla(\mathbf{w}))\big). \tag{26}$$

Equation (23) used (15); (24) used (18); (25) used (13) twice; and (26) collected terms and used (19). Substituting (26) into (22) and collecting terms yields

$$\nabla(\mathbf{q}) = \left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\mathbf{w}^T\mathbf{w}}\right)\big(\nabla(\mathbf{v}) - s\nabla(\mathbf{w})\big) - \frac{\big(\mathbf{w}\mathbf{q}^T\big)\nabla(\mathbf{w})}{\mathbf{w}^T\mathbf{w}}, \tag{27}$$

which, together with (20), completes the proof. □

Corollary 3.2. *With definitions as above, in a region where* $|\mathbf{v}| \neq 0$

$$\nabla\left(\frac{\mathbf{q}}{|\mathbf{v}|}\right) = \frac{1}{|\mathbf{v}|}\nabla(\mathbf{q}) - \frac{\mathbf{q}\mathbf{v}^T}{|\mathbf{v}|^3}\nabla(\mathbf{v}). \tag{28}$$

Proof. Apply (15) to get

$$\nabla\left(\frac{\mathbf{q}}{|\mathbf{v}|}\right) = \frac{1}{|\mathbf{v}|}\nabla(\mathbf{q}) - \frac{\mathbf{q}}{|\mathbf{v}|^2}\nabla(|\mathbf{v}|),$$

and use the identity $\nabla(|\mathbf{v}|) = \mathbf{v}^T\nabla(\mathbf{v})/|\mathbf{v}|$. □

The vector $\mathbf{q}$ can be, and effectively has been [18], used as the measure of nonparallelism (i.e., as the function $\mathbf{g}$ in (17)). However, Corollary 3.2 is motivated by the fact that

$\mathbf{q}/|\mathbf{v}|$ is dimensionless; its euclidean length is exactly $\sin\theta$, recalling that θ is the angle between $\mathbf{v}$ and $\mathbf{w}$. Since we are concerned with finding points where $\mathbf{g}$ in (17) is $\mathbf{0}$ without necessarily being at such a point, we would prefer that the gradient of our measure not be negative simply because $\mathbf{v}$ and/or $\mathbf{w}$ is getting smaller in magnitude. A function whose value is independent of the magnitudes of $\mathbf{v}$ and $\mathbf{w}$ is theoretically superior. We believe that (28) is the first workable expression for the gradient of a dimensionless measure of parallelism.

Combining (17), (21), and (28) and multiplying through by $|\mathbf{v}|$, we have a formula that we call `PVsolve` for $\delta\mathbf{x}$ such that $\mathbf{x}_0 + \delta\mathbf{x}$ is estimated to be a feature point (based on first-order terms). In this equation, all terms except $\delta\mathbf{x}$ are evaluated at $\mathbf{x}_0$

$$\mathbf{q} + \mathbf{P}_\mathbf{w}(\nabla(\mathbf{v}) - s\nabla(\mathbf{w}))\delta\mathbf{x} - \left(\frac{\mathbf{w}\mathbf{q}^T}{(\mathbf{w}^T\mathbf{w})}\nabla(\mathbf{w}) + \frac{\mathbf{q}\mathbf{v}^T}{(\mathbf{v}^T\mathbf{v})}\nabla(\mathbf{v})\right)\delta\mathbf{x} = \mathbf{0}. \tag{29}$$

Notice that the matrix multiplying $\delta\mathbf{x}$ is $n \times m$ and $\delta\mathbf{x}$ is an mD vector. Before looking at detailed solution procedures, we review the general structure of solutions of linear systems.

3.2 Homogeneous and Particular Solutions

The structure of solutions to linear systems, in general, is characterized by *homogeneous* solutions and *particular* solutions. The characterization encompasses square and nonsquare matrices, as well as singular and nonsingular matrices (only square nonsingular matrices have an inverse). We introduce the abbreviations

$$\mathbf{A} = (\nabla(\mathbf{v}) - s\nabla(\mathbf{w})), \tag{30}$$

$$\mathbf{B} = \frac{\mathbf{w}\mathbf{q}^T}{(\mathbf{w}^T\mathbf{w})}\nabla(\mathbf{w}) + \frac{\mathbf{q}\mathbf{v}^T}{(\mathbf{v}^T\mathbf{v})}\nabla(\mathbf{v}), \tag{31}$$

$$\mathbf{A}_P = \mathbf{P}_\mathbf{w}\mathbf{A} - \mathbf{B}, \tag{32}$$

so that the `PVsolve` equation (see (29)) becomes

$$|\mathbf{v}|\nabla\left(\frac{\mathbf{q}}{|\mathbf{v}|}\right)\delta\mathbf{x} = \mathbf{A}_P\,\delta\mathbf{x} = -\mathbf{q}. \tag{33}$$

We have n equations in m unknowns. If $m < n$, there may be no solutions. When there are solutions (for any m, n), we require

$$\mathbf{A}_P\delta\mathbf{x}_P = -\mathbf{q}, \tag{34}$$

$$\mathbf{A}_P\delta\mathbf{x}_H = \mathbf{0}. \tag{35}$$

Then solutions of (33) take the form $\delta\mathbf{x} = \delta\mathbf{x}_P + \gamma\delta\mathbf{x}_H$, where the scalar γ ranges over all reals. Solutions of (35) constitute the *null space* of $\mathbf{A}_P$. If $\mathbf{A}_P$ has rank m, its null space is the single point $\mathbf{0}$, and $\delta\mathbf{x}_P$ is unique if it exists. If $\mathbf{A}_P$ has rank less than m, its null space has positive dimension and $\delta\mathbf{x}_P$ is not unique.

Now we consider the special case $\mathbf{q} = \mathbf{0}$ in more detail. Note that this implies $\mathbf{B} = \mathbf{0}$ also. We define

$$\mathbf{A}_H = \mathbf{P}_\mathbf{w}\mathbf{A}. \tag{36}$$

Then, (33) reduces to

$$\mathbf{P}_\mathbf{w}\mathbf{A}\delta\mathbf{x}_H = \mathbf{A}_H\delta\mathbf{x}_H = \mathbf{0}, \tag{37}$$

and we call solutions to this equation the *homogeneous tracing solutions*. Recall that $\mathbf{A}_H$ is an $n \times m$ matrix. We make the following important observation.

Lemma 3.3 (Reduced rank principle). *The rank of $\mathbf{A}_H$ is at most $n - 1$.*

Proof. The rank of $\mathbf{P}_\mathbf{w}$ is exactly $n - 1$. □

An immediate consequence is that, if $\mathbf{x}_0$ is a feature point, and $n = m$, then the direction of the homogeneous tracing solutions is tangent to a feature curve passing through $\mathbf{x}_0$. Although it is possible that the rank of $\mathbf{A}_H$ is less than $n - 1$, Theisel et al. and others before them argue that this is an unstable condition and can be disregarded in practice (assuming that m is at least $n - 1$, as it is in all cases of interest here).

Another consequence applies if $m = n + 1$; in practice, time is the extra "space" dimension. Then the rank of $\mathbf{A}_H$ is at most $m - 2$ (stable cases are exactly $m - 2$). This implies that its null space is 2D and is tangent to a feature surface in mD passing through the feature point $\mathbf{x}_0$. The solutions of (37) define this 2D linear subspace.

In the third case of interest, $m = n - 1$, if $\mathbf{x}_0$ is a feature point, it is isolated in the stable cases. This applies to feature points restricted to cell faces (3D) or cell edges (2D).

Before looking at how to solve the various cases, we consider how (37) is related to previously published solutions.

3.3 Relationship to Other Tracing Formulations

Section 3.3 mathematically compares `PVsolve` ((29) or (33)) in the special case that $\mathbf{x}_0$ is a feature point with two previously published 3D tracing formulations. To accomplish this, we first need to analyze the formulas of Theisel et al. and find matrix expressions for their FFF when the vector fields are in 3D ($n = 3$). Their formulas use derivatives of the cross product, which are somewhat challenging to express with matrix algebra.

The key is to use the *cross-product matrix* representation. With the notation $w(i)$ for the ith component of $\mathbf{w}$, define

$$\chi_\mathbf{w} = \begin{bmatrix} 0 & -w(3) & w(2) \\ w(3) & 0 & -w(1) \\ -w(2) & w(1) & 0 \end{bmatrix}. \tag{38}$$

Then, for all $\mathbf{v}$, $\mathbf{w} \times \mathbf{v} = \chi_\mathbf{w}\mathbf{v}$, which is a 3D column vector by our convention.

A useful relationship between $\chi_\mathbf{w}$ and $\mathbf{P}_\mathbf{w}$, which is the projection matrix for $\mathbf{w}$, can be observed by noting that all columns of the projection matrix for $\mathbf{w}$ are orthogonal to $\mathbf{w}$. Therefore, the cross-product operation, applied to such columns, simply rotates by $\pi/2$ around $\mathbf{w}$ as an axis (denoted by $\mathbf{R}(\mathbf{w}, \pi/2)$) and scales by $|\mathbf{w}|$

$$\chi_\mathbf{w} = \chi_\mathbf{w}\mathbf{P}_\mathbf{w} = |\mathbf{w}|\mathbf{R}(\mathbf{w}, \pi/2)\mathbf{P}_\mathbf{w}. \tag{39}$$

Now, a matrix expression for the gradient of the cross product can be derived. The authors have not seen this identity in the literature, so it is stated as a lemma here.

TABLE 1
Homogeneous Tracing Equations for Three Methods

Method	Equation	Eq. no.
Sukharev *et al.*	$\mathbf{A}\,\delta\mathbf{x} = \mathbf{v}$	(5)
PVsolve($\mathbf{q} = \mathbf{0}$)	$\mathbf{P_w A}\,\delta\mathbf{x} = \mathbf{0}$	(37)
Theisel *et al.*	$\lvert\mathbf{w}\rvert\,\mathbf{R}(\mathbf{w}, \pi/2)\,\mathbf{P_w A}\,\delta\mathbf{x} = \mathbf{0}$	(44)

Lemma 3.4. *Let* $\mathbf{v}$ *and* $\mathbf{w}$ *be 3D vectors defined over an* m*D euclidean space. Then,*

$$\nabla(\mathbf{w} \times \mathbf{v}) = \chi_{\mathbf{w}} \nabla(\mathbf{v}) - \chi_{\mathbf{v}} \nabla(\mathbf{w}). \tag{40}$$

Proof. First, we observe the rule for differentiation with respect to a scalar variable, which is verified by applying the definitions

$$\frac{\partial}{\partial x}(\mathbf{w} \times \mathbf{v}) = \mathbf{w} \times \frac{\partial}{\partial x}\mathbf{v} - \mathbf{v} \times \frac{\partial}{\partial x}\mathbf{w} = \chi_{\mathbf{w}} \frac{\partial}{\partial x}\mathbf{v} - \chi_{\mathbf{v}} \frac{\partial}{\partial x}\mathbf{w},$$

with similar formulas for y and z (and t for the 4D case). Lining the results up in m columns gives (40). □

Returning to the problem addressed by Theisel et al., assuming we are at a feature point $\mathbf{x}_0$, we seek to solve

$$\nabla(\mathbf{w} \times \mathbf{v})_0 \delta\mathbf{x} = \mathbf{0}. \tag{41}$$

Using (40), this becomes

$$(\chi_{\mathbf{w}} \nabla(\mathbf{v}) - \chi_{\mathbf{v}} \nabla(\mathbf{w}))\delta\mathbf{x} = \mathbf{0}, \tag{42}$$

where all quantities except $\delta\mathbf{x}$ are evaluated at $\mathbf{x}_0$.

So far, this is a succinct version of the derivation in Theisel et al. But now we exploit the matrix notation to simplify further. We have $\chi_{\mathbf{v}} = s\chi_{\mathbf{w}}$ at $\mathbf{x}_0$, so (42) reduces to

$$\chi_{\mathbf{w}}(\nabla(\mathbf{v}) - s\nabla(\mathbf{w}))\delta\mathbf{x} = \chi_{\mathbf{w}}\mathbf{A}\delta\mathbf{x} = \mathbf{0}. \tag{43}$$

Using (39), this constraint becomes

$$|\mathbf{w}|\mathbf{R}(\mathbf{w}, \pi/2)\mathbf{P_w A}\delta\mathbf{x} = \mathbf{0}. \tag{44}$$

We are now ready to compare the equations of the various methods for homogeneous tracing solutions. The equations are collected in Table 1. We want to emphasize that the third equation is a mathematical representation of the matrix used by Theisel et al., chosen for analytical clarity. One should *not* infer that their method is more complicated or costly. The actual computation is based on cross products (see (1) and (43)), and has about the same cost as PVsolve($\mathbf{q} = \mathbf{0}$); the first equation in the table (see (5) and 6) should be faster than the others.

Suppose $\delta\mathbf{x}$ solves the first equation in the table. Then, due to collinearity of $\mathbf{v}$ and $\mathbf{w}$ at $\mathbf{x}_0$, $\mathbf{P_w v} = \mathbf{0}$, so the second and third equations hold. In the stable case for $n = m = 3$, the matrices in the second and third equations have rank 2 and their null space is 1D, so all solutions are collinear with the $\delta\mathbf{x}$ found with the first equation. Since $|\mathbf{w}|\mathbf{R}(\mathbf{w}, \pi/2)$ is nonsingular, the second and third equations have the same solutions, even in unstable cases, where the null space has dimension greater than one, and even if $m \neq 3$ ($n = 3$ is required for the cross products to be defined).

This completes the proof of mathematical equivalence of all three methods for stable cases *at feature points*. To summarize, we have derived a constraint (PVsolve, (29) or (33)) whose solutions in $\delta\mathbf{x}$ are such that $\mathbf{x}_0 + \delta\mathbf{x}$ is a feature point (to first-order terms), even if $\mathbf{x}_0$ is *not* a feature point. This constraint is valid for general n and m. We have analyzed the relationships of several methods for the special case that $n = m = 3$ and $\mathbf{x}_0$ is a feature point. We have shown that two previously published tracing methods are essentially equivalent to each other and are special cases of PVsolve (see (37)).

3.4 Solution Methods

We now turn to solution procedures for the various cases of PVsolve, (29) or (33). The most familiar case is when $\mathbf{A}_P$ is nonsingular, which implies that $\mathbf{x}_0$ is not a feature point, as discussed above. Then the solution is unique and is a Newton-Raphson step toward a nearby feature point

$$\delta\mathbf{x} = -\mathbf{A}_P^{-1}\mathbf{q}(\mathbf{x}_0). \tag{45}$$

This case requires $n = m$, but not necessarily $n = 3$. Convergence is guaranteed by repeating this step if the second-derivative tensor satisfies a definiteness condition, but the condition is too complicated to make checking practical.

A problematic case occurs when $m = n - 1$. When $n = 3$, this corresponds to finding a feature point in a plane, usually a cell face. Solutions are isolated points, so if $\mathbf{x}_0$ is a feature point, there is nothing to do. Otherwise the system is overdetermined and (in the stable cases) no solution exists. An approach that has been reported is to find a point that has the least error in some sense, then repeat the search from there. Previously reported procedures (for $n = 3$) minimize the magnitude of the cross product, based on a local linear model of it [13], [14], [19], [18]. A standard least-squares problem is solved [4], based on the matrix that multiplies $\delta\mathbf{x}$ in (42) (which is 3×2).

Equation (29) provides an alternative that deserves to be explored. The least-squares problem minimizes $|\mathbf{q}|$ instead of $|\mathbf{v} \times \mathbf{w}|$. Since $|\mathbf{v} \times \mathbf{w}| = |\mathbf{w}||\mathbf{q}|$, the two problems are related, but not equivalent.

Other cases are the underdetermined systems, either because the matrix has more columns than rows or is not full rank. Their solutions are most succinctly expressed using the *generalized cross product* [11, p. 700], [5] and the *adjugate matrix* [10], [17]. Both concepts are closely related to Cramer's rule. The adjugate is a kind of "surrogate inverse." For this discussion, let $\mathbf{M}$ be the matrix of the linear system.

The first important case is where $\mathbf{M}$ is square ($n = m$) and has rank $n - 1$. The obvious case of interest is $n = 3$, but we proceed generally, and we shall see later that $n = 4$ is also useful; in addition, $n = 2$ applies to 2D fields.

A handy mnemonic for 3×3 matrices may already be familiar to some readers as part of a rule for inverses: The jth *column* of the adjugate of $\mathbf{M}$ is the cross product of the *rows* of $\mathbf{M}$ indexed by $j + 1$ and $j + 2$, with indexes wrapping around.

The adjugate matrix of a square matrix $\mathbf{M}$ is written as $\mathrm{adj}(\mathbf{M})$ but we denote it as $\mathbf{M}^A$ for conciseness, following

Stewart [17]; it is defined as follows: Let $\mathbf{r}_i$ be *row* vectors representing the rows of $\mathbf{M}$. Then, the jth *column* of $\mathbf{M}^A$ is

$$\mathbf{M}^A_{*,j} = \det \begin{bmatrix} \mathbf{r}_1 \\ \cdots \\ \mathbf{r}_{j-1} \\ e_1, e_2, \ldots, e_m \\ \mathbf{r}_{j+1} \\ \cdots \\ \mathbf{r}_m \end{bmatrix}, \tag{46}$$

where e_i are treated symbolically (i.e., as uninterpreted symbols) for the computation of the determinant. This yields an expression $\mathbf{M}^A_{*,j} = (b_{1,j}e_1 + \cdots + b_{m,j}e_m)$ in which $b_{i,j}$ are real numbers. Then, e_i are treated as the standard basis vectors, so that $\mathbf{M}^A_{*,j} = [b_{1,j}, \ldots, b_{m,j}]^T$ is the final result.

Some important properties of the adjugate matrix are found in [10], [17]. Continuing the convention $\mathbf{M}^A = \mathrm{adj}(\mathbf{M})$, it is known that

$$\mathbf{M}\mathbf{M}^A = \mathbf{M}^A\mathbf{M} = (\det \mathbf{M})\mathbf{I}, \tag{47}$$

$$(\mathbf{M}^T)^A = (\mathbf{M}^A)^T, \tag{48}$$

$$(\mathbf{M}\mathbf{N})^A = \mathbf{N}^A\mathbf{M}^A. \tag{49}$$

When $\det \mathbf{M} \neq 0$, then $\mathbf{M}^A$ can be used to compute $\mathbf{M}^{-1}$, through (47). However, $\mathbf{M}^A$ is sometimes useful even when $\det \mathbf{M} = 0$. It is worth noting that (1) and (6) have succinct expressions using adjugates and (48): $\mathbf{C}^A\mathbf{a}$ and $\mathbf{A}^A\mathbf{v}$, respectively.

The expression on the right-hand side of (46), possibly negated, is also known as the generalized cross product. In this case, it denotes $\pm \mathsf{X}_m(\mathbf{r}_{j+1}, \ldots, \mathbf{r}_m, \mathbf{r}_1, \ldots, \mathbf{r}_{j-1})$. We use ℓ to denote the last row index of $\mathbf{M}$ (m if indexing is one-based). If m is even and $\ell - j$ is odd, the minus sign must be used; in all other cases, the plus sign is correct. Note that the generalized cross product of mD vectors requires $m-1$ operands. It is the usual cross product when $m = 3$, and shares many properties of the usual cross product when $m \neq 3$.

Two other important properties of the adjugate matrix are relevant to the problem we address here, but we have *not* seen these in the literature.

Lemma 3.5. *Let* $\mathbf{M}$ *be* $n \times n$ *and let* $\mathbf{M}^A = \mathrm{adj}(\mathbf{M})$.

1. *When the rank of* $\mathbf{M}$ *is* $n-1$, *any nonzero column of* $\mathbf{M}^A$ *lies in the 1D null space of* $\mathbf{M}$; *specifically, if removing row* i *from* $\mathbf{M}$ *would leave* $n-1$ *linearly independent rows, then column* i *of* $\mathbf{M}^A$ *is nonzero and provides a homogeneous solution of* $\mathbf{M}\delta\mathbf{x} = \mathbf{0}$.
2. *When the rank of* $\mathbf{M}$ *is* $n-2$ *or less,* $\mathbf{M}^A$ *is identically zero.*

Proof. For part 1, by hypothesis, after removing row i, leaving $n-1$ linearly independent rows, it is possible to remove some column, say j, leaving a nonsingular matrix. The determinant of this matrix (possibly negated) is $\mathbf{M}^A_{j,i}$. Since $\det(\mathbf{M}) = 0$, (47) implies that $\mathbf{M}\mathbf{M}^A_{*,i} = \mathbf{0}$.

For part 2, by hypothesis, after removing *any* row i, the remaining $n-1$ rows are *not* linearly independent, so $\mathbf{M}^A_{j,i} = 0$ for all choices of j. □

Thus, in the case $n = m = 3$ where $\mathbf{x}_0$ is a feature point, any nonzero cross product of two rows of $\nabla(\mathbf{q})_0$ (see (37)) provides a vector in the tangent direction. Using the largest such vector should give the best numerical accuracy.

Another important property of the adjugate matrix is easy to derive, but we have not seen it in the literature. Informally stated, eigenvectors of nonzero eigenvalues transfer between a matrix and its adjugate.

Lemma 3.6. *Let* $\mathbf{M}$ *be* $n \times n$ *and let* $\mathbf{M}^A = \mathrm{adj}(\mathbf{M})$. *Let the eigenvalues of* $\mathbf{M}$ *be indexed as* λ_j, $j = 1, \ldots, n$.

1. *If* ξ *is an eigenvector of* λ_j *for* $\mathbf{M}$, *then* ξ *is an eigenvector of* μ_j *for* $\mathbf{M}^A$, *where*

$$\mu_j = \prod_{1=1, i \neq j}^{n} \lambda_i. \tag{50}$$

2. *If* ξ *is an eigenvector of* μ *for* $\mathbf{M}^A$, *where* $\mu \neq 0$, *then* ξ *is an eigenvector of* $\lambda = (\det \mathbf{M})/\mu$ *for* $\mathbf{M}$.

Proof (Sketch). The lemma is trivial when the rank of $\mathbf{M}$ is n (inverse exists) or is $n-2$ or less ($\mathbf{M}^A = \mathbf{0}$). The remaining case, rank $n-1$, involves eigendecomposition into real Jordan form [6], [21]. For any real $\mathbf{M}$, there is a matrix $\mathbf{T}$ with determinant 1 such that $\mathbf{J} = \mathbf{T}^{-1}\mathbf{M}\mathbf{T}$ is in real Jordan form, and every real eigenvector of $\mathbf{M}$ appears as a column of $\mathbf{T}$ (not necessarily unit length). But, $\mathbf{T}^{-1} = \mathbf{T}^A$ by (47), so $\mathbf{J}^A = \mathbf{T}^{-1}\mathbf{M}^A\mathbf{T}$ by (49). Since $\mathbf{J}$ has the same eigenvalues as $\mathbf{M}$ and these include precisely one eigenvalue equal to zero, it suffices to prove the lemma for $\mathbf{J}$ and $\mathbf{J}^A$, and this is straightforward with Lemma 3.5 and standard linear algebra. □

The practical importance of Lemma 3.6 for our purposes is this: If $\mathbf{M}$ is diagonalizable, as it is in stable cases, and has one eigenvalue that is much smaller than the others, then the eigenvector for the small eigenvalue varies continuously and is well approximated by large columns of $\mathbf{M}^A$; it is not necessary to decide whether the small eigenvalue is precisely zero. To relate this to integrating in an FFF, if there is a nearby point on a feature curve, the appropriate matrix $\mathbf{M}$ has a zero eigenvalue on that curve and the eigenvector is in the tangent direction. If $\mathbf{M}$ and $\mathbf{M}^A$ are evaluated nearby, $\mathbf{M}$ will not have an eigenvalue that is precisely zero, but the direction given by the eigenvector corresponding to the small eigenvalue will be close to the tangent direction.

Now we turn to the case in which $n = m - 1$. This is typically a case with time as the "extra" space dimension. The main case of interest is $n = 3$, $m = 4$.

First, we cover the case that $\mathbf{x}_0$ is *not* a feature point, and $\mathbf{M}$ has full rank. Equation (29) is nonhomogeneous since $\mathbf{q} \neq \mathbf{0}$. The complete solution involves both a particular part and a homogeneous part. We can define $\mathbf{M}'$ with $n' = n + 1$ rows by adding a row of zeros to the bottom of $\mathbf{M}$. This reduces the homogeneous problem for $\mathbf{M}'$ to the one just considered (Lemma 3.5, part 1), with $n' = m$ and a matrix of rank $n' - 1$. Only column m of the adjugate matrix $\mathbf{M}^A$

need be computed (the rest are identically zero), and the homogeneous solution is (any scalar multiple of) the generalized cross product of the original $m-1$ rows of $\mathbf{M}$, i.e., $\mathbf{X}_m(\mathbf{M}_{1,*}, \ldots, \mathbf{M}_{m-1,*})$.

To find a *particular* solution, discard a column of $\mathbf{M}$ that leaves n linearly independent columns (in stable cases, the first n columns are linearly independent), set the corresponding element of $\delta\mathbf{x}$ to zero, and solve the resulting $n \times n$ system for the remaining elements of $\delta\mathbf{x}$.

It is important to note that the above procedure is *useless at a feature point* because $\mathbf{M}$ has rank $n-1$, so $\mathbf{M}'$ has rank $n'-2$ (at most), and the adjugate matrix is identically zero by Lemma 3.5, part 2. Thus, the FFF proposed by Theisel and Seidel [20, (9)] is seen to be identically zero, using (44). (Theisel et al. note that this proposed FFF "appears" to be identically zero. Our lemma confirms this.)

Finally, we consider the case that $n = m-1$ and $\mathbf{x}_0$ *is* a feature point. In the stable case, as discussed above, $\mathbf{M}$ has rank $n-1 = m-2$. Because $\mathbf{q} = \mathbf{0}$ in (29), it suffices to consider homogeneous solutions. Choose two columns j and k such that deleting them would leave $m-2$ linearly independent columns. Normally, $j = m$, the column associated with time, is an acceptable choice, and k can be one of $m-1$ and $m-2$. However, different choices might be more robust, numerically.

This time, we define $\mathbf{M}'$ with $n' = n+1$ rows by adding a row $\mathbf{e}_j^T$ (zeros, except 1 in column j) to $\mathbf{M}$. This again produces an $\mathbf{M}'$ with rank $n'-1$, so the solution given above provides one homogeneous solution, whose jth component is zero. (See (46), but column m of $\mathbf{M}^A$ need not be computed, as it is zero.) If $j = m$, this is just the fixed-time homogeneous tracing solution.

For the final step, we define $\mathbf{M}''$ using k instead of j in the preceding paragraph, and get a homogeneous solution whose kth component is zero. Assuming a stable case, the two solutions are linearly independent and span the null space. If it is desirable to have orthogonal homogeneous solutions for some reason, just use the Gram-Schmidt procedure [4].

3.5 A Predictor-Corrector Procedure for Tracing

A predictor-corrector procedure is easily formulated, using (18)-(29), and Section 3.4. To simplify the notation, we use $\mathbf{Q}$ for $\mathbf{A}_P$, as defined in (32), and we use $\mathbf{S}$ for $\mathbf{A}_P{}^A$, the adjugate of $\mathbf{Q}$ (46). For $n = m = 3$, with the convention that the cross product of row vectors is a row vector, we have

$$\mathbf{S} = \begin{bmatrix} \mathbf{Q}_{2,*} \times \mathbf{Q}_{3,*} \\ \mathbf{Q}_{3,*} \times \mathbf{Q}_{1,*} \\ \mathbf{Q}_{1,*} \times \mathbf{Q}_{2,*} \end{bmatrix}^T. \tag{51}$$

That is, cross products of rows of $\mathbf{Q}$ give columns of $\mathbf{S}$. When $\det \mathbf{Q} \neq 0$ and $\mathbf{Q}$ is sufficiently well conditioned, we use $\delta\mathbf{x} = -(\mathbf{S}/\det \mathbf{Q})\mathbf{q}$. Otherwise, all three columns of $\mathbf{S}$ are collinear (within numerical tolerance) and one of the largest magnitude is selected to provide the *direction* for $\delta\mathbf{x}$. The length of $\delta\mathbf{x}$ in the latter case is chosen heuristically.

Starting from a *seed point* $\mathbf{x}_0$, calculate $\mathbf{Q}$, $\mathbf{S}$, and $\det \mathbf{Q}$ according to those equations. If $\mathbf{x}_0$ is a feature point, $\det \mathbf{Q}$ will be 0 or very close. In this case, use the maximum-magnitude column of $\mathbf{S}$ in (51) as the tangent direction. Take a "predictor" step in this direction to $\mathbf{x}_1^{(0)}$, which is slightly off the feature curve. The length of this step is chosen heuristically. Now use (21), (45), and (51) to take corrector steps until $\det \mathbf{Q}$ is sufficiently close to 0

$$\begin{aligned} \delta\mathbf{x}^{(n)} &= -\left(\mathbf{Q}(\mathbf{x}_1^{(n)})\right)^{-1} \mathbf{q}(\mathbf{x}_1^{(n)}), \\ \mathbf{x}_1^{(n+1)} &= \mathbf{x}_1^{(n)} + \delta\mathbf{x}^{(n)}. \end{aligned} \tag{52}$$

Call the final point $\mathbf{x}_1$. This is numerical root finding in 3D and subject to all the pitfalls typical of Newton-Raphson in higher dimensions, but under reasonable circumstances $\mathbf{x}_1$ is on the feature curve. When (45) is used, $\delta\mathbf{x}^{(n)}$ may need to be clamped to stay within the cell in which interpolation is being done. The procedure is repeated to trace a polyline $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ... Experimental results are reported in Section 6.

4 Illustrative Analytical Example

To see whether the predictor-corrector procedure, using the extra term in (27) worked in practice, we tested it on pairs of linear vector fields. We considered this a reasonable first test because nonlinear vector fields are nearly linear at the small scales used in numerical procedures. We compared our results with the analytical solution [21]. We note that the equations previously published for tangent-based tracing [19], [18] traced this curve very accurately using Runge-Kutta 4/5 with tight error tolerances (10^{-8} or 10^{-12}, see Section 6 for configuration details).

First, we consider the following 2D flow:

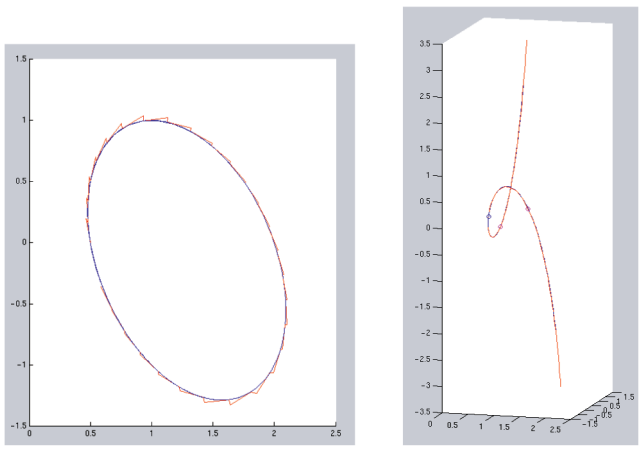
$$\begin{aligned} \mathbf{v} &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \text{crit. pt. } \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\ \mathbf{w} &= \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -2 \\ 0 \end{bmatrix} \quad \text{crit. pt. } \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \end{aligned}$$

This paper's predictor-corrector method, `PVsolve`, is shown in Fig. 2a. We see that the corrector-moves converge to the correct feature curve. The picture shows both predictor points and corrector points to illustrate the method. Only the final corrector point of a sequence is output by `PVsolve` as a feature point.

Turning to 3D linear fields, we traced the following:

$$\begin{aligned} \mathbf{v} &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 1 & -1 & 1 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \quad \text{crit. pt. } \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \\ \mathbf{w} &= \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix} \quad \text{crit. pt. } \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}. \end{aligned}$$

As seen in Fig. 2b, `PVsolve` tracks the analytical solution to subpixel accuracy. (The same accuracy is achieved by published methods using Runge-Kutta 4/5 with tight error tolerances.) This view is looking primarily toward positive x. The three circles show where the curve crosses $z = 0$. Looking toward negative z, the projection onto x-y is the same as the 2D example (if traced to $z = \infty$ and $z = -\infty$), i.e., in the limit, the projection completes one oval and approaches the limiting point $(2, -1)$ from both sides.



(a) (b)

Fig. 2. `PVsolve` moves with step .20. Blue: analytical solution curve. Red: trace for 63 points counting both predictor and corrector moves. (a) 2D, all moves. (b) 3D, converged moves only.

We observed that the 2D flows required only one corrector move for each predictor move, even though the predictor move was fairly large, at 0.2. The threshold for convergence was $\det \mathbf{Q}/|\mathbf{Q}|_F < 10^{-10}$ (F denotes Frobenius matrix norm). However, the more complicated feature curve associated with the 3D flows required an average of three corrector-moves for each predictor-move. Other predictor-corrector schemes that have been proposed specify that the corrector-moves are in a plane normal to the predictor-move. This is definitely not a requirement for this paper's method.

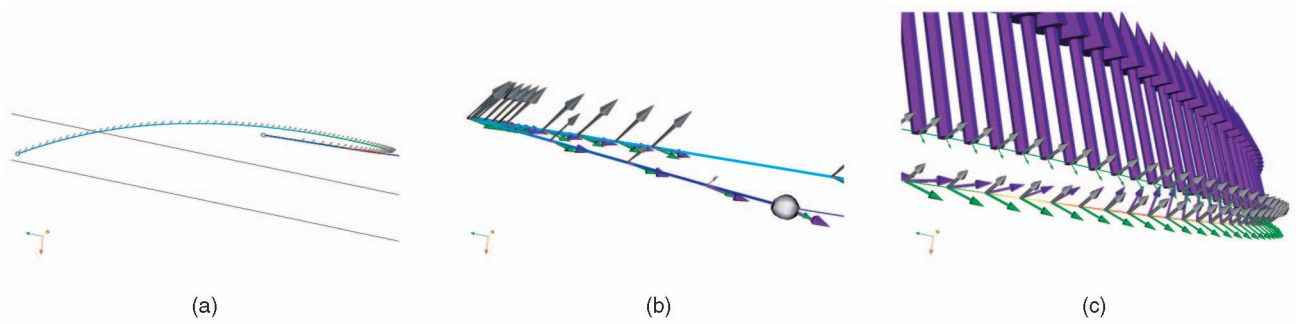
5 Feature Flow Field Goes Astray

This section reports on a surprising pattern of behavior that we observed using the FFF method on several curvilinear data sets. Actually, the analysis in Section 3 explains the behavior. By FFF in this section, we mean the equations given by Theisel et al. [19] for steady flows, integrated with `ode45`, a Runge-Kutta 4/5 procedure supplied by *matlab*. The tolerance options were set for maximum accuracy. Additional details of the experimental setup are given in Section 6. The online supplement contains additional images related to this section.

In the majority of cases, FFF traced feature curves very accurately. However, we noticed a strange phenomenon in occasional cases while examining the numerical values generated for the curve. In these cases, the trace would "head for" a region where the two vector fields $\mathbf{v}$ and $\mathbf{w}$ were perpendicular. Recall that θ denotes the positive angle between these two fields. The trace seemed to be acting "deliberately," in the sense that the $\sin\theta$ values climbed steadily until they exceeded 0.99 for several points, then receded steadily. Such occurrences would be very unlikely due to noise-like errors; in such cases, we would expect $\sin\theta$ to vary more or less randomly.

This phenomenon is illustrated in Fig. 3. The left image shows the overview. The FFF curve starts from the seed point indicated by the sphere on the right, makes a barely visible jog to the left, then quickly turns right, continues through a section where its color changes from cyan to red and back to green, then makes a sharp turn and heads toward the left. The `PVsolve` curve starts from the same seed point and simply proceeds to the right, its blue color indicating $\sin\theta$ near zero. The FFF curve varies in color, reflecting various angles between $\mathbf{v}$ and $\mathbf{w}$; blue denotes $\sin\theta = 0$ and red is $\sin\theta = 1$. The gray arrows show the cross product (about every eight points in the overview, and every point in the closeups). The `PVsolve` curve stays mostly on track, has one bad point in the middle, reaches a cell face to the right of the picture, and continues into the adjoining cell. The FFF curve exits at an incorrect face and does not continue, because the vectors there are not sufficiently parallel.

The center image is an extreme closeup, which shows that the cross products grow steadily from zero for several steps during the initial jog to the left, after which time they stay fairly constant in both magnitude and direction, as indicated by the overview image. At this zoom factor, this image also shows that the initial direction of the FFF curve (thicker) is actually going left, while the `PVsolve` (thinner) is going right, so the two curves really have no part in common. The violet arrows are $\mathbf{v}$ (velocity) and the green arrows are $\mathbf{w}$ (curl of velocity). The center image shows them almost parallel in the blue region. The right image shows them becoming perpendicular, then returning to a smaller angle (green is 30° or $\sin\theta = 0.5$). The magnitudes are scaled differently in the two closeups and are different between the violet and green arrows, to improve clarity of direction at the important places. Actually, $\mathbf{v}$ is orders of magnitude smaller than $\mathbf{w}$ throughout.



(a) (b) (c)

Fig. 3. Feature curve traced by FFF, color mapped to $\sin\theta$, going from blue for zero to red for one, on the Post data set (see Section 6). Gray arrows are the cross product of $\mathbf{v}$ and $\mathbf{w}$. (a) Overview. (b) Beginning of trace. (c) The $\mathbf{v}$ and $\mathbf{w}$ fields become perpendicular where the curve is red.

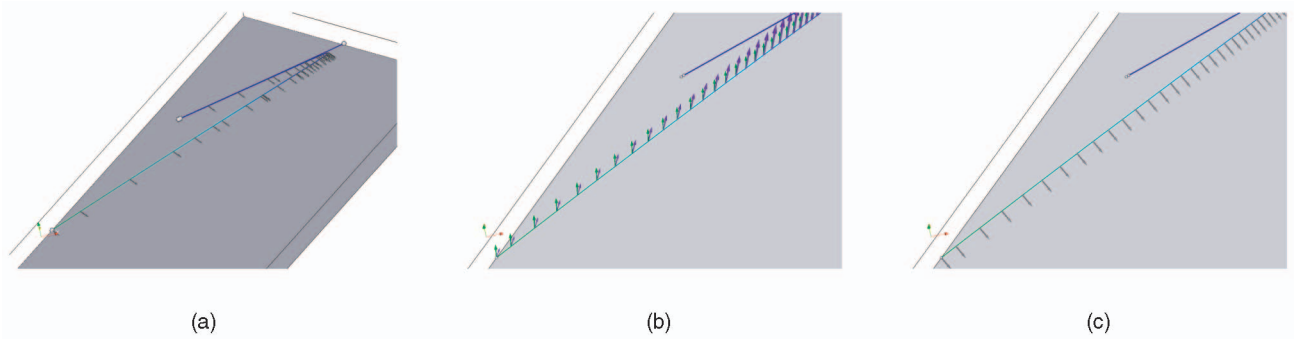


Fig. 4. (a) Overview. (b) $\mathbf{v}$ and $\mathbf{w}$ arrows on FFF curve. (c) Cross product shown by gray arrows.

Holding the cross product constant is consistent with the description of FFFs, in general [20]. Indeed, the tangent direction is defined by the criterion that the gradient of the cross product, informally stated, is "zero in this direction." Ideally, it would be held constant at zero. However, in this case, although it starts at zero, the fields appear to vary by higher order polynomials than Runge-Kutta can compensate for, and the cross product grows to a slightly nonzero value. This happens in essentially every cell. In most cases, it is harmless.

In this case, an unintended consequence of the FFF strategy materializes. As the right closeup shows, $\mathbf{v}$ is decreasing in magnitude along the curve. How can the cross product be held constant if $|\mathbf{v}|$ is decreasing? The answer is: *increase the angle between* $\mathbf{v}$ *and* $\mathbf{w}$!

So the system is forced into a region of larger θ in its effort to keep the cross product constant along the curve. The question arises, since the system is *not* computing at a point where the cross product is zero, is the computed tangent direction still a "direction of zero gradient?" More precisely, does the gradient matrix have an eigenvector close to the computed tangent direction, and is the associated eigenvalue very small? The analysis in terms of the adjugate matrix (see discussion after Lemma 3.6 in Section 3) answers the question affirmatively. The experimental data confirm that the FFF system is quite successful at keeping the cross product constant in both magnitude and direction.

Another instance of the same phenomenon is shown in Fig. 4. In the overview, FFF and `PVsolve` start along the same curve, proceeding from the seed point (marked by the middle sphere) toward the upper right, but FFF changes direction, while `PVsolve` continues to the cell face. The color of the lines are mapped to $\sin\theta$ as before: blue is zero, green is 0.5. Gray arrows indicate the cross product on the FFF curve. Before making the sharp turn, the blue color informs us that the magnitude of the cross product is quite small. However, FFF is formulated to keeping a gradient of the cross product constant, and is therefore concerned with both the direction and magnitude. We observe that FFF not just preserving the magnitude; it also "wants to" preserve the direction. So when the direction on the true curve varies, FFF is forced off. The closeups show that in order to keep the cross product constant, there are regions where θ is increasing, as needed to balance magnitude decreases in $\mathbf{v}$ and/or $\mathbf{w}$. On the the other hand, `PVsolve`, which does not use the cross product at all, concentrates on keeping $\sin\theta$ small.

Although such instances are not frequent, over the whole Post data set, the FFF method traced to nearly 18,000 points where $\sin\theta \geq 0.90$, i.e., θ is between 60° and 120°. This is about 7 percent of the 242,000 traced points. In contrast, `PVsolve` arrived at only 16 such points out of about 102,000 traced.

6 Results

Our tracing method, `PVsolve`, was implemented in *matlab* and integrated into a prototype parallel-vector program under development by Jeff Sukharev, for which results on regular grids have been reported [18]. We thank the author for making his code available. Tests were performed on a 2.6-GHz x86 Linux platform. We are not aware of published times or measures of accuracy for other methods that analyze parallel vectors, so we can only present our own times and measures. The tests reported here are intended to serve mainly as a proof of concept, rather than an exhaustive evaluation.

For the curvilinear simulations, we set the threshold for convergence of corrector steps to $\det \mathbf{Q}/|\mathbf{Q}|_F^3 < 10^{-8}$ (subscript F denotes Frobenius matrix norm). We set a limit of 20 corrector steps for each predictor step to protect against nonconvergence. All steps were clamped not to move out of the current curvilinear cell, and corrector steps were clamped to be the same order of magnitude as the preceding predictor step. The average number of corrector steps per predictor step was 1.72, with 95 percent of the calls taking one or two corrector steps, while 2.5 percent limited out at 20.

We used several well-known curvilinear data sets distributed by NASA Ames in the *plot3d* format (see Table 2). In all cases the two fields tested for collinearity were $\mathbf{v}$ for velocity and $\mathbf{w}$ for vorticity (curl of velocity), as calculated by the NASA Ames program, *FAST*.

First, we present some evaluation data comparing `PVsolve` with the FFF method that uses the homogeneous formulas from Theisel et al. discussed in Section 3, driven by an explicit Runge-Kutta (4,5) solver supplied in *matlab*. This solver, `ode45`, is sophisticated and highly optimized [15]. Then we show some visualizations of the parallel-vector curves obtained by our procedures in context with other visual information.

The statistics are based on all feature curves discovered, sometimes called *raw* feature curves. For particular applications, it is usually appropriate to add some additional

TABLE 2
PVsolve Tracing Times (CPU) and Numbers of Points for Steady-Flow Curvilinear Data Sets

Dataset	Grid points	Seed Points	Traced Points	Minutes
Blunt Fin	40,960	331	39,444	5.20
Post	109,744	763	109,762	18.78
Delta-40	211,680	1377	186,026	62.72

criteria to *filter out* feature curves that are not of interest. For exploratory purposes, we applied one such filter in some of the runs. Let us call a seed point *swirling* if the velocity gradient has two complex eigenvalues at that point. Then a *swirling feature curve* is defined to be a feature curve generated from a swirling seed point. Such a curve might also pass through other seed points that are not "swirling." By applying this *swirling filter* eliminated about 70 percent of the raw feature curves, fairly consistently across the data sets studied. The curves used for illustrations in Section 5 qualified as swirling feature curves.

6.1 Statistical Evaluation and Comparisons

The columns of Table 3 require some explanation. The first column describes the procedure. The FFF/ode45 procedure was tested with three levels of error tolerance, as described under the table. The second column shows the total length of traced curves measured in computational space (i.e., each cell side is of unit length). This choice is based on the fact that cells are smaller in the important regions, so measuring physical length would overweight curves in less important regions. The third column is how many distinct curves were traced. The fourth column measures efficiency, with arc length being the same as the second column; this measure avoids rewarding a program that finishes faster but produces less product.

The last two columns contain error measures. The fifth column is the average of $\sin\theta$, where θ is the angle between $\mathbf{v}$ and $\mathbf{w}$. The sixth column is the fraction of cells in which the traced curve had $\theta > 1^\circ$ upon exit from the cell. The curve always begins the cell with $\theta < 1^\circ$, and when it begins at a seed point in the face of that cell, $\theta < 10^{-10}$ degrees. If $\theta > 1^\circ$ upon exit, tracing along this curve is discontinued. Hence, different methods have differing total arc lengths in part due to discontinuing at different cell faces.

Since tighter error tolerances have produced greater accuracy for FFF/ode45 in the table, the natural question is whether more is better. We tried 10^{-13} but the program slowed way down and failed by exceeding our limit of 3,000 integration steps in a single cell. So, it appears that 10^{-12} is about the tightest we can go. Unless stated otherwise, FFF/ode45 will refer to runs using the 10^{-12} tolerances.

The error columns show that PVsolve is more accurate than FFF/ode45 by a factor of about 50 using average $\sin\theta$ as the criterion. Column six shows a factor of about two difference in failure rates. Column four shows that the gains come at a modest price in time.

We gathered similar statistics for the Blunt Fin and Delta-40 data sets, but only using the most accurate settings for FFF/ode45. The results are shown in Table 4. These data sets show generally the relationship between the methods as the Post, but both methods achieve less accuracy. We also checked the statistics for "swirling" feature curves and found that failure rates were about half, while average error decreased from 15 percent to 35 percent.

TABLE 3
Comparison of FFF/ode45 with Several Configurations and PVsolve on the Post Curvilinear Data Set

Method	Tot. Arc Length	Num. of Curves	Arc Length per CPU Sec.	Avg. $\sin\theta$	Failure Rate
ode45 (default)	2634	370	30.6	0.0511	0.080
ode45 (10^{-8})	2540	349	17.8	0.0409	0.052
ode45 (10^{-12})	2409	334	5.0	0.0212	0.025
PVsolve	4065	291	3.6	0.0004	0.012

ode45 configurations: "default" means 10^{-3} for *RelTol*, 10^{-6} for *AbsTol*. Otherwise, both parameters are the number shown in parentheses.

See Section 6.1 for discussion.

TABLE 4
Comparison of FFF/ode45 and PVsolve on the Bluntfin and Delta-40 Curvilinear Data Sets

Method	Tot. Arc Length	Num. of Curves	Arc Length per CPU Sec.	Avg. $\sin\theta$	Failure Rate
Bluntfin					
ode45 (10^{-12})	764	221	2.9	0.0621	0.165
PVsolve	1414	194	4.5	0.0297	0.100
Delta-40					
ode45 (10^{-12})	3079	554	4.0	0.0213	0.065
PVsolve	6339	516	1.7	0.0025	0.041

See Section 6.1 for discussion.

6.2 Pictures

Figs. 5, 6, 7, 8, 9, and 10 show the feature curves extracted using PVsolve on the three steady flow, curvilinear data sets. Since the choice and formulation of the parallel vectors, using velocity and vorticity, are intended to identify vortex cores, we also seeded streamlines in the regions where vortex core structures are known to exist. All streamlines are integrated using ParaView's [9] implementation of Runge-Kutta 4/5 with a tolerance of 10^{-6} for maximum error. The results confirm the locations of the vortex structures in the Blunt Fin and Delta-40 data sets, but the major vortical streamlines do not correspond to any feature curve found in the Post data set. This raises the possibility that parallelism of velocity and vorticity is not a suitable criterion for vortex cores in the Post data set.

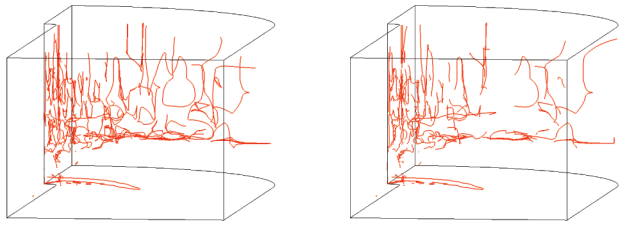


Fig. 5. Overview images of the Blunt Fin showing extracted feature curves in red using (a) PVsolve with 194 lines and (b) FFF/ode45 with 221 lines.

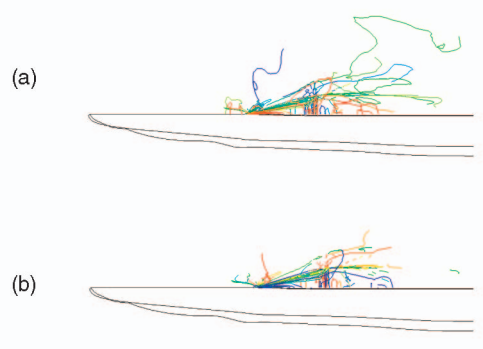


Fig. 6. Overview image of the Delta-40 extracted feature curves using PVsolve with 516 lines (a), and FFF/ode45 with 554 lines (b). Curves are colored differently to show crowding of lines at the nose.

In general, the feature curves look reasonable—with two caveats. First, they may not identify vortex cores in all cases because other flow phenomena might cause these fields to be parallel. It is possible to apply a postprocessing operation to filter out these spurious or nonphysical feature curves by using some of the application-dependent criteria suggested by Peikert and Roth [13]. In this paper, we decided not to apply any filtering operation to overview pictures (see Figs. 5, 6, and 7) since the focus of the paper is on the ability of `PVsolve` to locate curves where the underlying vector fields are parallel (i.e., a mathematical rather than a physical description). In addition, presenting all the raw feature curves facilitates future comparisons with this work. Thus, while the parallel-vector formulation used in the examples in this section is motivated by finding the vortex cores, the raw feature curves in the figures include those that are not physically realistic candidates for vortex cores.

A second caveat is that there are numerous ways to formulate what constitutes a vortex core, even with parallel vectors. The feature curves, while faithful to the requirement of having both $\mathbf{v}$ and $\mathbf{w}$ parallel to each other, may not be an ideal representation of the true vortex cores. In fact, it is possible that a true vortex core is entirely missed by this formulation. Research on vortex core extraction is active and challenging, but is beyond the scope of this paper.

One may also note from the overview images in Figs. 5, 6, and 7 that while `PVsolve` generally produced less feature lines than `FFF/ode45`, the feature lines are longer and have higher connectivity.

Blunt fin. Fig. 8 shows the feature curves extracted by `PVsolve`. The prominent (half) horseshoe vortex core feature curve in red is easily seen on the left image. This particular feature is also correctly traced by `FFF/ode45` and the method of Sukharev et al. The middle image shows only the "swirling" feature curves, as defined at the beginning of Section 6, while the right image shows the "raw" feature curves, i.e., all of them.

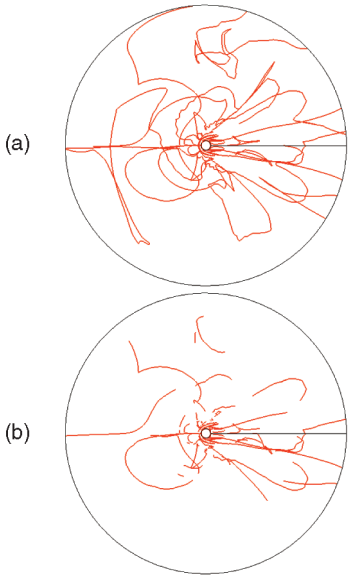


Fig. 7. Overview images of extracted feature curves from Post in red using (a) PVsolve with 291 lines and (b) FFF/ode45 with 334 lines.

Delta wing. The next data set is the Delta wing at a 40° angle of attack. Fig. 9 shows the "swirling" feature curves extracted by `PVsolve` in red against (half of) the wing in magenta. As with the other data sets, these comprise about 30 percent of the "raw" feature curves. Streamlines are seeded near the nose and also near the leading edge of the wing. We see the main vortex core off the wing as well as one along the leading edge of the wing. The streamlines are colored by integration time and goes from green to red to improve the contrast. Flow reversal is apparent in the main core off the wing where one can observe some of the yellowish streamline heading back toward the nose. The image on the bottom shows a closeup of the leading edge where one can observe a similar behavior.

Post. The next data set is the flow past a cylinder and is referred to as the Post data set. Unlike the other two data sets, this one simulates an incompressible fluid flow; i.e., the

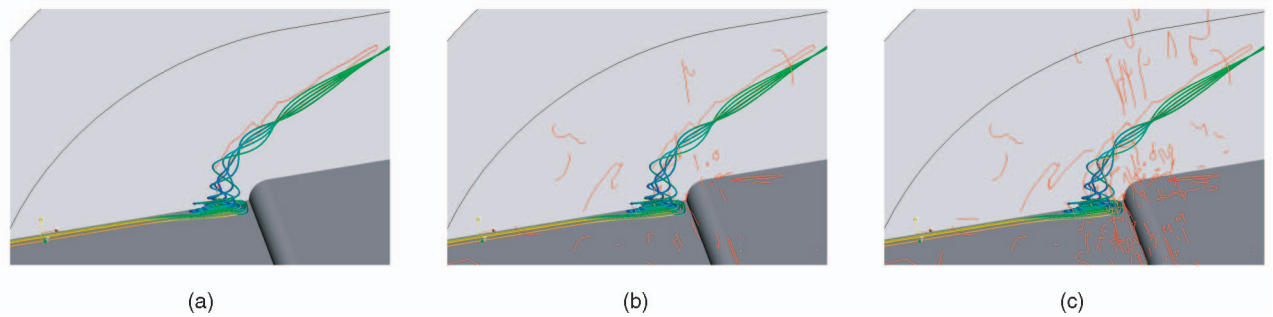


Fig. 8. Feature curves extracted using this paper's method (red) and streamlines illustrating a well-known vortex in the Blunt Fin. Streamlines are colored by velocity magnitude. (a) The dominant vortex feature curve. (b) Feature lines with complex eigenvalues of velocity gradient. (c) All "raw" feature lines.

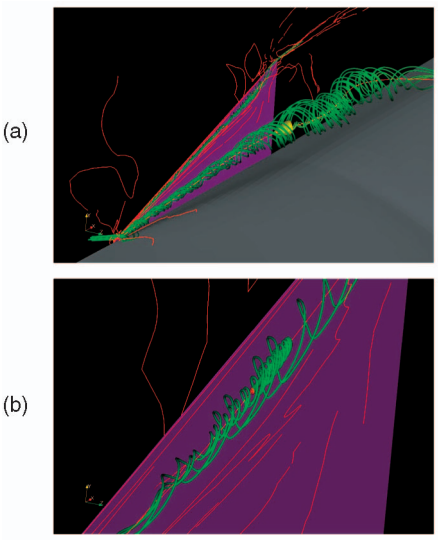


Fig. 9. The Delta Wing at a 40° angle of attack. This data set models half of the wing shown in magenta, the extracted feature curves in red, and streamlines colored by integration time. (a) Overview image showing the main vortex structure off the wing and a smaller one along the leading edge of the wing. Note reverse flow as streamlines turn from green to yellow in the larger vortex. (b) Closeup of the vortex along the leading edge. The closeup shows reverse flow in the smaller vortex also.

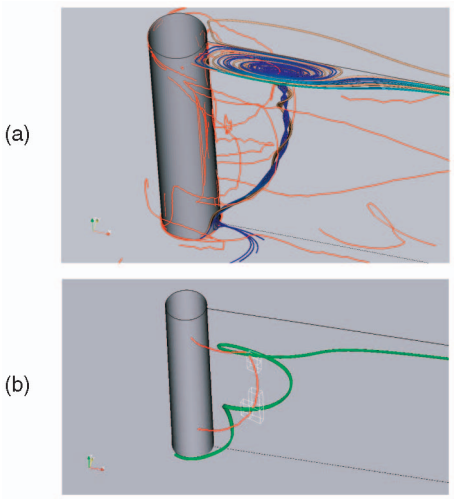


Fig. 10. Feature curves in red, streamlines colored by their seeding location. (a) The swirling pattern shows one of two prominent vortex structures in this data set. There is no feature curve detected within this vortex structure using the criterion of parallel velocity and vorticity. Tan streamlines originated at a feature curve, but that curve did not continue into the vortex region using any of the tracing methods studied. (b) Analysis of the supposedly main feature curve shows that streamlines seeded along this curve do not join up with the main vortical structure on the top image.

density is constant throughout, although pressure varies. The curvilinear grid wraps around on itself on the plane directly downwind of the post. This plane also forms a plane of symmetry of the extracted feature curves. Note that the flow is not perfectly symmetric, but the major features at the scale of the images in Fig. 10 can be considered symmetric. The image on the top shows how the streamlines, seeded from various selected places in the data set, come together to form one of two, very distinct, vortex core structures just downstream of the post. The other, very similar vortex core structure would be on the opposite side of the plane of symmetry, and is not shown.

What is clearly noticeable is that there is no feature curve ("raw" or "swirling") running through this vortex structure. The main feature curve that is supposed to be the vortex core is extracted and analyzed further on the bottom image. The boxes indicate places where seed points were found and used to trace the rest of this feature curve. We verified that the $\mathbf{v}$ and $\mathbf{w}$ vectors along this curve are indeed parallel (very small $\sin\theta$). Furthermore, seeding along this vortex core, e.g., in the vicinity of the seed point, does not produce streamlines that join up with the vortical flow. This is not a failing of `PVsolve` (indeed FFF/`ode45` produced substantially the same results), but rather points to the inadequacy of using the parallel velocity and vorticity criterion to find vortex cores, at least in the Post data set.

7 Conclusion and Future Work

We have presented a new formulation, `PVsolve`, and applied it to tracing parallel vectors. It is more robust in some ways than previous tracing methods. The main property of `PVsolve` is a correction step that allows the trace to home in on the feature curve, i.e., where the two vector fields are parallel. The formulation also generalizes the two previous formulations: FFF [20], [19] and analytical tangents [18]. Although they both required the seed point as well as points along the trace to be on the feature line (i.e., $\mathbf{q} = \mathbf{0}$), `PVsolve` handles the case where $\mathbf{q} \neq \mathbf{0}$.

Initial experiments on linear fields indicated that the corrector steps converged rapidly to the correct feature curve with fairly large predictor steps. Another observation was that the corrector steps are *not* in the plane orthogonal to the predictor step, which differs from previously proposed predictor-corrector schemes for tracking vortex cores and parallel vectors. Future work should address how to determine appropriate step sizes for predictor steps. Furthermore, the current formulation is based on first-order approximations. Using higher order information, accumulated through a number of probes, would be another avenue for improvement.

The method was tested on three well-known curvilinear fluid-flow data sets and compared to results of `FFF` traced using `ode45` in *matlab*. An electronic supplement in the digital library has additional images and code.[1] These are steady-flow simulations. Statistics show that `PVsolve` achieved between 2 and 50 times lower average error, based on $\sin\theta$, with a modest increase in compute time. Features were identified in the regions where vortex cores are known to exist based on other techniques, such as streamlines. However, we only checked the results visually and are not sure how accurate the correspondence is. The experiments turned up a surprising phenomenon, discussed in Section 5: Once a small error is established in FFF, the system undergoes "contortions" to preserve that error, occasionally leading it away from the correct feature curve.

A topic for future work is to look at using the cross product, rather than the dimensionless projection vector, as

1. See also http://avis.soe.ucsc.edu/PVsolveSupplemental.

the quantity for root finding, as suggested by an anonymous referee. A quick test showed that the method works (as everyone expected), but in view of the findings in Section 5, hidden problems related to scale may surface.

An unexplored alternative for seed point location that deserves future investigation is to use the (nonlinear) closed-form solution for the locus of parallel vectors in a general pair of 3D linear (more precisely, affine) vector fields [21]. (When the linear fields have a common critical point, the locus is in the direction of a certain eigenvector, emanating from the critical point [14].)

The model of an FFF [19] is appealing, but does not provide for correcting drift, which is inevitable in numerical procedures. This paper's method, in its current form, is more appropriately viewed as root finding, rather than streamline integration. Future work should study how to combine these ideas to obtain a *smooth* FFF in which the directions at points somewhat away from the correct feature curve tend to send the trace closer to that curve.

ACKNOWLEDGMENTS

This work was partially supported by the UCSC/Los Alamos Institute for Scalable Scientific Data Management (ISSDM). The authors thank Holger Theisel and Tino Weinkauf for facilitating the comparison of their results with those from their earlier works; Ronny Peikert, Martin Roth, and Jeffrey Sukharev for their codes. They also thank the anonymous reviewers for their feedback and helpful suggestions.

REFERENCES

[1] D.C. Banks and B.A. Singer, "Vortex Tubes in Turbulent Flows: Identification, Representation, Reconstruction," *Proc. IEEE Visualization Conf. '94,* pp. 132-139, 1994.
[2] D.C. Banks and B.A. Singer, "A Predictor-Corrector Technique for Visualizing Unsteady Flow," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 151-163, June 1995.
[3] C. Garth, R.S. Laramee, X. Tricoche, J. Schneider, and H. Hagen, "Extraction and Visualization of Swirl and Tumble Motion from Engine Simulation Data," *Topology-Based Methods in Visualization,* H. Hauser, H. Hagen, and H. Theisel, eds., pp. 121-135, Springer, 2007.
[4] G.H. Golub and C.F. Van Loan, *Matrix Computations,* third ed. Johns Hopkins University Press, 1996.
[5] A.J. Hanson, *Geometry for N-Dimensional Graphics,* P. Heckbert, ed., pp. 149-170, Academic Press, 1994.
[6] M.W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, 1974.
[7] R. Hooke and T.A. Jeeves, "'Direct Search' Solution of Numerical and Statistical Problems," *J. ACM,* vol. 8, no. 2, pp. 212-229, 1961.
[8] J. Jeong and F. Hussain, "On the Identification of a Vortex," *J. Fluid Mechanics,* vol. 285, pp. 69-94, 1995.
[9] Kitware, *ParaView Guide, Kitware,* Version 3, http://www.paraview.org, Feb. 2008.
[10] M. Marcus and H. Minc, *Introduction to Linear Algebra.* MacMillan, 1969.
[11] W.S. Massey, "Cross Products of Vectors in Higher Dimensional Euclidean Spaces," *Am. Math. Monthly,* vol. 90, pp. 697-701, 1983.
[12] G.M. Nielson and I.-H. Jung, "Tools for Computing Tangent Curves for Linearly Varying Vector Fields over Tetrahedral Domains," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 4, pp. 360-372, Oct.-Dec. 1999.
[13] R. Peikert and M. Roth, "The Parallel Vector Operator—A Vector Field Visualization Primitive," *Proc. IEEE Visualization Conf. '99,* pp. 263-270, 1999.
[14] M. Roth, "Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization," PhD thesis, ETH Zurich, Inst. of Scientific Computing, Diss. ETH No. 13673, 2000.
[15] L.F. Shampine and M.W. Reichelt, "The Matlab ODE Suite," *SIAM J. Scientific Computing,* vol. 18, pp. 1-22, 1997.
[16] S. Stegmaier, U. Rist, and T. Ertl, "Opening The Can of Worms: An Exploration Tool for Vortical Flows," *Proc. IEEE Visualization Conf. '05,* pp. 463-470, 2005.
[17] G.W. Stewart, "On the Adjugate Matrix," *Linear Algebra and Its Applications,* vol. 283, pp. 151-164, 1998.
[18] J. Sukharev, X. Zheng, and A. Pang, "Tracing Parallel Vectors," *Proc. SPIE Visual Data Analysis and Exploration,* 2006.
[19] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Extraction of Parallel Vector Surfaces in 3D Time-Dependent Fields and Application to Vortex Core Line Tracking," *Proc. Visualization Conf. '05,* pp. 631-638, 2005.
[20] H. Theisel and H.-P. Seidel, "Feature Flow Fields," *Proc. Joint Eurographics—IEEE TCVG Symp. Visualization (VisSym'03),* pp. 141-148, 2003.
[21] A. Van Gelder, "Report on Relaxed Jordan Canonical Form for Computer Animation and Visualization," technical report, Univ. of California, Santa Cruz, http://www.cse.ucsc.edu/~avg/Papers/rjcfTR.pdf, Jan. 2009.
[22] T. Weinkauf, "Extraction of Topological Structures in 2D and 3D Vector Fields," PhD thesis, Otto von Guericke Univ. of Magdeburg, Germany, 2008.
[23] T. Weinkauf, J. Sahner, H. Theisel, H.-C. Hege, and H.-P. Seidel, "A Unified Feature Extraction Architecture," *Active Flow Control,* R. King, ed., pp. 119-133, Springer, 2007.

Allen Van Gelder received the BS degree in mathematics from Massachusetts Institute of Technology and the PhD degree in computer science from Stanford University in 1987. He is currently a professor of computer science at the University of California, Santa Cruz. He has worked in the areas of computer animation, scientific visualization, algorithms, and logic. He is a member of the IEEE Computer Society.

Alex Pang received the BS degree in industrial engineering from the University of the Philippines, and the MS and PhD degrees in computer science from the University of California at Los Angeles, in 1984 and 1990, respectively. He is currently a professor of computer science at the University of California, Santa Cruz. He has worked in the areas of comparative and uncertainty visualization, and flow and tensor visualization. He is a senior member of the IEEE and a member of the IEEE Computer Society.