Two–dimensional global manifolds of vector fields

Bernd Krauskopf
Dept. of Eng. Maths.
University of Bristol
Bristol BS8 1TR
U.K.
B.Krauskopf@bristol.ac.uk

Hinke Osinga
Control & Dyn. Systems
Caltech 107–81
Pasadena, CA 91125
U.S.A.
hinke@cds.caltech.edu

November 1998

Abstract

We present an algorithm for computing two-dimensional stable and unstable manifolds of three-dimensional vector fields. The main idea is to grow the manifold in concentric (topological) circles. Each new circle is computed as a set of intersection points of the manifold with a finite number of planes perpendicular to the last circle. Together with a scheme for adding or removing such planes this guarantees the quality of the mesh representing the computed manifold.

As examples we compute the stable manifold of the origin spiralling into the Lorenz atractor, and an unstable manifold in Arneodo's system converging to a limit cycle.

1 Introduction

Vector fields are the mathematical models of choice in numerous applications; see for example [7, 16] and further references therein. The most basic question is to understand the dynamics of a given fixed vector field. The phase space is organized by the spine or skeleton of the dynamics, which consists of all equilibria and other invariant sets, like periodic orbits, together with their stable and unstable manifolds. These stable and unstable manifolds are global objects forming boundaries between regions with different qualitative behavior, and their intersection can lead to complicated dynamics and chaos.

Hence, knowing the spine means knowing the fate of the orbit of any initial condition. Consequently, finding the stable and unstable manifolds is an important issue. It is generally not possible to find stable and unstable manifolds other than via numerical computation.

One-dimensional stable and unstable manifolds consist of two distinct orbits that can be computed by integrating two suitable initial conditions. However, higher-dimensional stable and unstable manifolds consist of infinitely many orbits, and their computation remains a challenge; see Section 2 for an overview of other methods. In this paper we consider the lowest dimensional such case, namely two-dimensional stable and unstable manifolds of equilibria in a three-dimensional state space. To be more concrete we consider a three-dimensional vector field

$$\dot{x} = f(x), \tag{1}$$

where $x \in \mathbf{R}^3$ and $f : \mathbf{R}^3 \mapsto \mathbf{R}^3$ is sufficiently smooth. The dynamics is determined in this case by the stable and unstable manifolds of equilibria and periodic orbits of saddle type. Recall that the stable and the unstable manifold of a saddle point x_0 are defined as

$$\begin{aligned} W^s(x_0) &= \{x \in \mathbf{R}^3 \mid \lim_{t\to\infty} \phi^t(x) = x_0\}, \\ W^u(x_0) &= \{x \in \mathbf{R}^3 \mid \lim_{t\to-\infty} \phi^t(x) = x_0\}, \end{aligned}$$

where ϕ^t is the flow of (1). The Stable and Unstable Manifold Theorem [14] states that in a neighborhood of x_0 there exist local stable and unstable manifolds $W^s_{\mathrm{loc}}(x_0)$ and $W^u_{\mathrm{loc}}(x_0)$, which are tangent to the stable and unstable eigenspaces $E^s(x_0)$ and $E^u(x_0)$ of the Jacobian $Df(x_0)$ of (1), respectively. Either the stable or the unstable manifold of any saddle point x_0 is two-dimensional. For convenience we discuss the case of a two-dimensional unstable manifold $W^u(x_0)$ in the sequel.

Our algorithm grows $W^u(x_0)$ in circles starting with an initial circle on $E^u(x_0)$. Its important feature is that it computes the two-dimensional unstable manifold as a mesh of a prescribed quality. For each mesh point r on the last circle we do the following. We define a two-dimensional plane through r perpendicular to the last circle, and determine a new point on $W^u(x_0)$ in this plane at a suitable distance from r. Finding this new point amounts to solving a boundary value problem, which is done by a shooting appraoch. The algorithm works when x_0 has eigenvalues with largely differing modulus, as well as in the case of complex conjugate eigenvalues. As a particular feature we detect convergence to an attractor, which allows us to compute the entire unstable manifold in case it is compact.

The organization of the paper is as follows. We start with an overview of other methods in Section 2. We explain the algorithm in Section 3, and illustrate its performance with two examples in Section 4. Section 5 gives a summary of the results and discusses the generalizability of the algorithm.

2 Other methods

There are three different ideas to compute a two-dimensional unstable manifold for vector fields. In some sense the most straightforward idea, which is also our approach, is to start with a small circle of points on $E^u(x_0)$ around x_0 and grow the manifold as a one-parameter family of (topological) circles. The integration time provides the natural parameter. The main problem is that the circles typically do not grow uniformly in all radial directions. Since one can only evolve a finite number of points on any given circle one quickly encounters the problem that points drift apart, leading to a mesh of poor quality.

Johnson *et al.* [10] tackle this problem by integrating each of a finite number of nicely distributed points on a circle until a prescibed arclength is reached. This defines a new circle of points. In a second step extra points are added if necessary by using interpolation, after which the points are redistributed equally spaced along the piecewise linear circle to again obtain a nice distribution of points on the next circle. The procedure is repeated until a sufficient piece of the manifold has been obtained. Since the dynamics on the manifold controls the distance between the points on the new circle before the redistribution of mesh points, not much can be said about the accuracy of the interpolated points.

In the methods of Guckenheimer and Worfolk [8, 17] divergence between points in the integration is avoided by rescaling the vector field. In this way it can be achieved that the tangential component is practically zero at each point along a circle. This changes the dynamics, but not $W^u(x_0)$ as a set. Then all these points are integrated until a prescribed distance is reached to obtain the next circle. Again, a sufficient piece of the manifold can be grown by repeating this procedure. Their method requires that the vector field always has a radial component pointing outward with respect to the last circle.

Doedel [6] computes two-dimensional manifolds with a very different method using continuation. To start he computes an orbit of a prescribed arclenth (or arclength times integration time) by integrating as initial condition a point on a ray through x_0 on $E^u(x_0)$ very close to x_0. This orbit is then continued with the boundary value solver of AUTO [5], where the angle of the ray is the continuation parameter. In this way, the orbit sweeps over $W^u(x_0)$, which gives very accurate results. A possible disadvantage is that this method produces an abundance of meshpoint near x_0.

Finally, there is the approach of considering the time t map of the flow of (1) for some fixed $t > 0$, and use an algorithm designed for maps. This is done, for example, in [12]. Another interesting method for discrete systems is the one by Dellnitz and Hohmann [3, 4] who cover the manifold with three-dimensional boxes. They start with a covering of $W^u_{\text{loc}}(x_0)$ with sufficiently small boxes which they obtain with a subdivision method. In a second step the covering of $W^u_{\text{loc}}(x_0)$ is extended to obtain a covering of $W^u(x_0)$ in a particular window of interest. A two-dimensional approximation of $W^u(x_0)$ can be obtained by connecting the center points of the boxes in the covering with triangles. The accuracy of the computation is directly related to the diameter of the boxes, so that a large number of boxes may be required. We remark that, in general, using the time t map insted of the vector field itself leads to longer computations.

3 The algorithm

In this section we describe how to compute the two-dimensional unstable manifold $W^u(x_0)$ of a saddle point x_0 of the vector field (1). This means that we consider the case that one eigenvalue λ^s of x_0 is real and negative, and the other two eigenvalues λ^u_1 and λ^u_2 have positive real parts (and may be real or complex). We start with a linear approximation of the local unstable manifold $W^u_{\text{loc}}(x_0)$ and use the fact that the forward orbit of any point on $W^u_{\text{loc}}(x_0)$ lies on the global manifold $W^u(x_0)$. It is known [14] that $W^u(x_0)$ is tangent at x_0 to the plane $E^u(x_0)$ spanned by the (generalized) eigenvectors of λ^u_1 and λ^u_2. The starting data for our algorithm is a discrete circle in $E^u(x_0)$, by which we mean a finite set of points on a small circle of radius δ about x_0.

Algorithm GLOBALIZE

Input:	f,	right-hand-side of vector field;
	x_0,	saddle point of f;
	C_r, δ,	finite mesh on circle $C(x_0, \delta)$ in $E^u(x_0)$;
	Δ,	guess for distance between circles;
	L_{arc},	arclength to be computed;
Output:	mesh on $W^u(x_0)$ of arclength approximately L_{arc} from x_0	

`Band`	**array** [0,. . . ,3][$\|C_r\|$] of Point;
	Comment: representation for $\widehat{C}_p$, $\widehat{C}_r$, and $\widehat{C}_b$
`Leaf`	**array** [$\|C_r\|$] of Point;
`Delta`, `ARC`	**real**;
`Admissible`	**boolean**;

Begin

`Band[0]` $= x_0$; `Band[1]` $= C_r$;

    Comment: initially $\widehat{C}_p$ is circle with 0 radius

Print(`Band[0]`, `Band[1]`);

    Comment: data is stored in file

`ARC` $= \delta$;

while `ARC` $< L_{arc}$ **do**

  for $r \in C_r$ **do**

    `Leaf[r]` $= r_{right} - r_{left}$;

    Comment: Determine linear foliation $\{\mathcal{F}_r\}_{r \in C_r}$

    `Band[2][r]` = BVP(`Band[1]`, `Leaf[r]`, `r`, `Delta`);

  `Admissible` = ACCURACY(`Band`);

  if `Admissible` **then**

    `ARC` = `ARC` + `Delta`;

    `Band[0]` = `Band[1]`; `Band[1]` = `Band[2]`;

    Print(`Band[0]`, `Band[1]`);

  adjust `Delta`;

End.

Figure 1: *The algorithm* GLOBALIZE *in pseudo-code. The procedure* BVP *is described in Section 3.1, and the procedure* ACCURACY *is outlined in Section 3.2.*

The global manifold $W^u(x_0)$ is generated by computing more and more discrete circles, each of which consists of a finite number of mesh points. The manifold $W^u(x_0)$ is then represented as the triangulation given by these mesh points. Hence, adding a discrete circle corresponds to adding a band of triangles to the computed piece of $W^u(x_0)$.

Let C_r denote the last computed discrete circle of mesh points, and let $\widehat{C}_r$ denote the continous circle obtained by linearly connecting the mesh point in C_r. The key step is to add a new discrete circle, denoted by C_b, at a particular distance Δ from C_r. The problem is to ensure that the mesh points of C_b are nicely distributed. To achieve this, we define a linear foliation $\{\mathcal{F}_r\}_{r \in C_r}$, that is, a set of (half)planes $\mathcal{F}_r$, transversal to the last circle $\widehat{C}_r$. We now consider the one-dimensional intersection curves of $W^u(x_0)$ with $\{\mathcal{F}_r\}_{r \in C_r}$. The choice of $\{\mathcal{F}_r\}_{r \in C_r}$ has to be such that locally the intersection of $W^u(x_0)$ with each leaf $\mathcal{F}_r$ is a curve of sufficient arclength; compare [12]. In each leaf $\mathcal{F}_r$ we find a point in $W^u(x_0)$ at distance Δ from the base point r by solving a boundary value problem. This set of points forms the new discrete circle C_b, which indeed lies at distance Δ from C_r with nicely distributed mesh points. More details of this procedure are given below in Section 3.1; see also Figure 1.

If mesh points come too close to each other, or move apart too much we update the mesh, which is discussed in Section 3.2. The algorithm also detects and deals with convergence to an attractor, and this is explained in Section 3.3.

3.1 Adding a new circle

As the first step we need to find a foliation $\{\mathcal{F}_r\}_{r \in \widehat{C}_r}$ such that $W^u(x_0)$ intersects each leaf $\mathcal{F}_r$, $r \in C_r$, in a curve locally near $\widehat{C}_r$. A plane in three-dimensional space is uniquely defined by a normal vector, and we define $\mathcal{F}_r$ by its normal vector

$$(r_{right} - r) + (r - r_{left}) = r_{right} - r_{left}\,,$$

where r_{left} and r_{right} are the neighboring mesh points on the left and right of r, respectively. Note that this normal vector can be considered as tangent to C_r at r.

The next step is to find a point $b_r \in W^u(x_0) \cap \mathcal{F}_r$ at a prescribed distance Δ from r. The choice of Δ is influenced by the local curvature of $W^u(x_0)$.

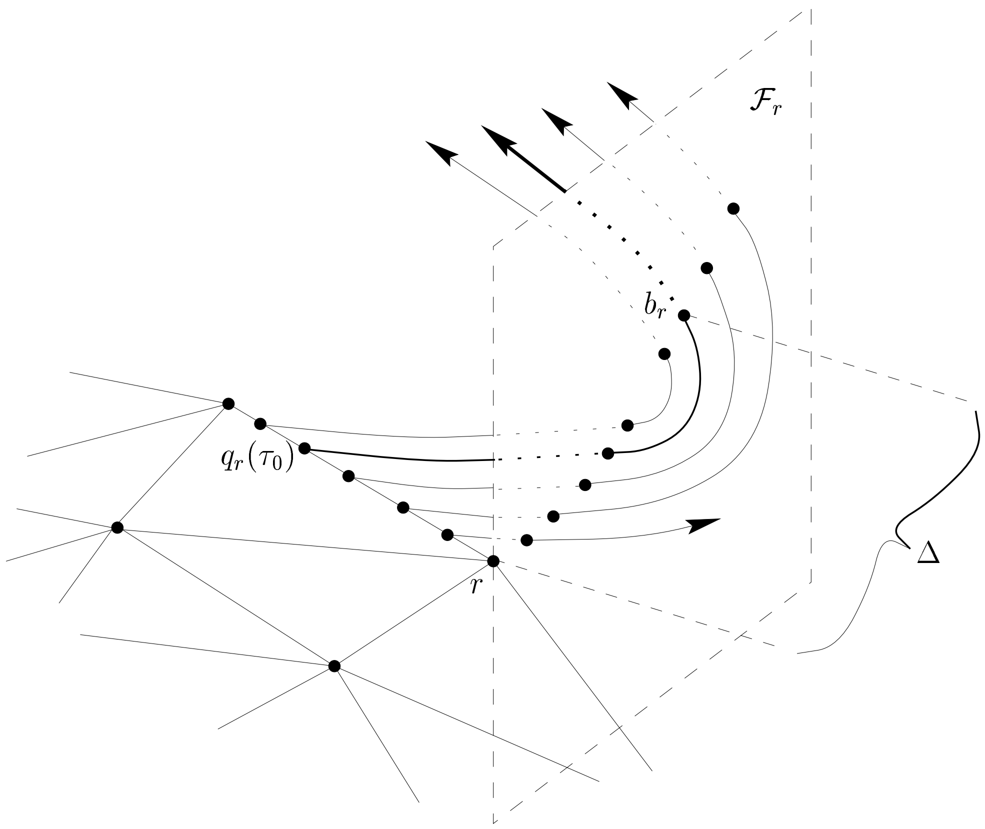


Figure 2: *The next point b_r in the plane $\mathcal{F}_r$ through r is found by continuation in τ of orbits $\{ \phi^t(q_r(\tau)) \mid t \in [0, \tau] \}$ starting on the last circle and ending in $\mathcal{F}_r$. The point $q_r(\tau_0)$ is uniquely defined as the point on the last circle for which $b_r = \phi^{\tau_0}(q_r(\tau_0))$ lies at distance Δ from r.*

In fact, Δ is a guess and it is checked at a later step whether this guess is in accordance with the required accuracy; see Section 3.2.

The key idea is that the next point $b_r \in \mathcal{F}_r$ can be found by continuing the following boundary value problem until the required distance Δ is reached. We require an orbit $\{ \phi^t(q_r(\tau)) \mid t \in [0, \tau] \}$ satisfying the boundary conditions

$$\left\{ \begin{array}{lcl} \phi^0(q_r(\tau)) & \in & \widehat{C}_r, \\ \phi^\tau(q_r(\tau)) & \in & \mathcal{F}_r. \end{array} \right. \tag{2}$$

For convenience we denote the initial condition in $\widehat{C}_r$ by $q_r(\tau)$, and the final point in $\mathcal{F}_r$ by $b_r(\tau) = \phi^\tau(q_r(\tau))$. The trivial solution $q_r(\tau) = r$ and $b_r(\tau) = r$ for $\tau = 0$ satisfies the boundary condition (2). We continue this solution in the continuation parameter τ while we monitor the distance

$$\| b_r(\tau) - r \| . \tag{3}$$

Note that this distance is zero for $\tau = 0$. The continuation stops when for some (first) τ_0 we have $\| b_r(\tau_0) - r \| = \Delta$. We set $b_r = b_r(\tau_0)$ and are done; see Figure 2.

In practice, we accept the point $b_r(\tau_0)$ if $\| b_r(\tau_0) - r \| \in [(1+\varepsilon)\Delta, (1+\varepsilon)\Delta]$ for some prespecified ε. In the examples of Section 4 we used $\varepsilon = 0.01$. This continuation part of the algorithm is performed in the procedure BVP in Figure 1. In the present implimentation we use a shooting method as a boundary value solver.

Remark 1 *Because we specified a maximal integration time for shooting, we may "run out" of integration time before we reach distance Δ. In particular, close to x_0 and, in case $W^u(x_0)$ converges to an attractor, near the attractor this may force a smaller Δ than necessary for the required accuracy; see Section 3.3.*

After the computation of C_b, we need to check whether the band of triangles between C_r and C_b accurately approximates $W^u(x_0)$. It is possible that $\widehat{C}_b$ lies too far from $\widehat{C}_r$, or that we need more mesh points in C_b. This is described in the next section.

3.2 Mesh adaptation

Our algorithm automatically updates the mesh after each step. The following is taken into consideration to determine whether the mesh is still accurate. First we check whether the distance Δ from $\widehat{C}_b$ to $\widehat{C}_r$ is sufficiently small. It may be that $\widehat{C}_b$ should be closer to $\widehat{C}_r$ in order to get an accurate approximation of $W^u(x_0)$. Next, we check whether $\widehat{C}_b$ satisfies the conditions for the mesh quality. If necessary, extra points are added, or removed. The mesh adaptation is done in the procedure ACCURACY in Figure 1.

After performing the step in the previous Section, both circles $\widehat{C}_r$ and $\widehat{C}_b$ consist of the same number of mesh points. We form lines from points in C_r to corresponding points in C_b. If these lines accurately follow the one-dimensional curves $W^u(x_0) \cap \{\mathcal{F}_r\}_{r \in C_r}$ locally near $\widehat{C}_r$, then we accept Δ

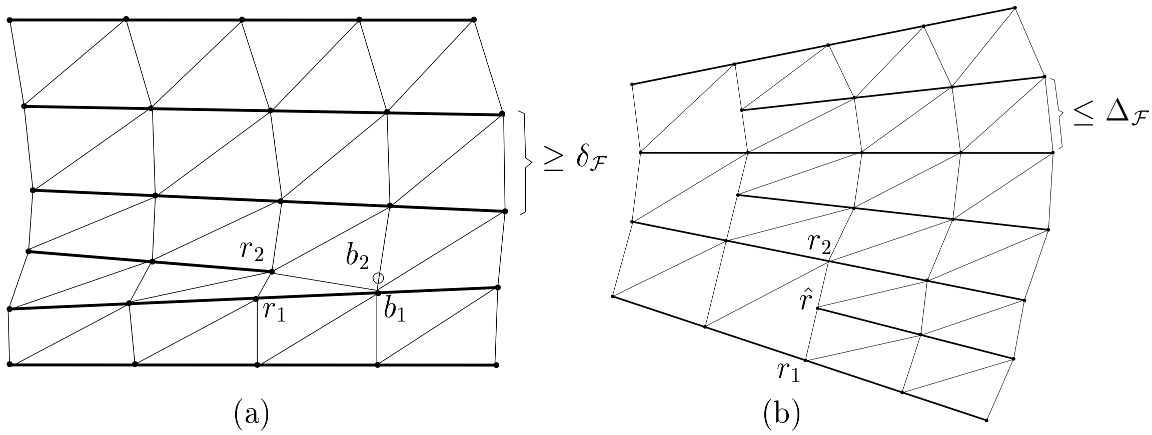


Figure 3: *Mesh adaptation by removing (a) and adding (b) leaves.*

as a good estimate. Otherwise, we reject C_b, decrease Δ, and repeat the step in Section 3.1. The accuracy of one-dimensional curves is controlled using the technique described in [13], which is based on controling the angle between three successive points together with the product of this angle and the distance between the last two of the three points; compare also [9]. Our next guess for Δ is based on these considerations as well. Since the angle is defined using three successive points, we need to keep track of the circle before $\widehat{C}_r$ in the data structure; see also Figure 1, where this circle is called C_p. Note that initially this circle has radius 0.

Once we have accepted $\widehat{C}_b$ as a new circle at a proper distance from $\widehat{C}_r$, we check how well $\widehat{C}_b$ approximates $W^u(x_0)$. The accuracy of $\widehat{C}_b$ is controlled by the distance between neighboring mesh points. We prespecify minimal and maximal distances between two neighboring mesh points, denoted by $\delta_{\mathcal{F}}$ and $\Delta_{\mathcal{F}}$, respectively. Let r_1 and r_2 be two neighboring mesh points in C_r, and let b_1 and b_2 denote the corresponding points in C_b.

It is most important that b_1 and b_2 are not too close to each other, since then, in the next step, the leaves through these points might cross at a distance less than Δ from b_1 or b_2. When the mesh points are not properly ordered, the algorithm will be confused when it computes the next circle. Hence, if $\| b_2 - b_1 \| < \delta_{\mathcal{F}}$, we wish to delete a mesh point. Therefore, we set $b_2 = b_1$, and drop b_2 as a mesh point in the next step; see also Figure 3 (a).

If $\| b_2 - b_1 \| > \Delta_{\mathcal{F}}$, then the mesh is not accurate enough. We wish to add an extra point to C_b. For this, we first add the point

$$\hat{r} = \frac{1}{2}(r_2 + r_1)$$

to C_r. We next define $\mathcal{F}_{\hat{r}}$ and compute $b_{\hat{r}}$ by solving the corresponding boundary value problem; see Figure 3 (b).

3.3 Convergence to an attractor

In the event that $W^u(x_0)$ is compact, it may not be possible to find points at distance Δ, because the manifold has finite arclength. However, the arclength generally differs from leaf to leaf. Therefore, when we detect convergence, we allow different Δ-steps in different leaves. This means that we let Δ depend on the mesh points $r \in C_r$.

For a particular leaf $\mathcal{F}_r$ we find $q_r(\tau)$ and $b_r(\tau) = \phi^\tau(q_r(\tau))$ during the continuation of the boundary value problem. Recall that we are monitoring the distance $\| b_r(\tau) - r \|$ until we find the point $b_r(\tau_0)$ for which this distance is Δ. If the manifold converges, we may have that $\| b_r(\tau) - r \|$ is always smaller than Δ, even for $\tau \to \infty$. If this is the case, we allow for different Δ-steps in different leaves.

In practice, we put a maximum $\tau_{\max}$ on τ. If we find that $\| b_r(\tau) - r \| < \Delta$ for all $0 < \tau \leq \tau_{\max}$, we take the best candidate, that is, we set $b_r = b_r(\tau_{\max})$, such that

$$\| b_r - r \| = \Delta_r < \Delta,$$

and Δ_r is as large as possible. This way, we assure a relatively smooth decrease and increase of Δ_r along $\widehat{C}_b$. Note that the orbits of points close to x_0 grow slowly, so that they may not be of sufficient arclength after this maximal integration time; compare Remark 1. In this case, we decrease Δ and insist on having the same distance to C_r for all points in C_b, so that the mesh stays nice. We only allow Δ to vary in different leaves after a certain arclength of the manifold has been computed.

The algorithm detects convergence in a leaf $\mathcal{F}_r$ if the best possible new point b_r lies at distance Δ_r smaller than a prescribed minimum. Subsequently, the algorithm stops computing new points in this leaf. If convergence is detected in all leaves, the algorithm stops altogether and reports convergence to an attractor.

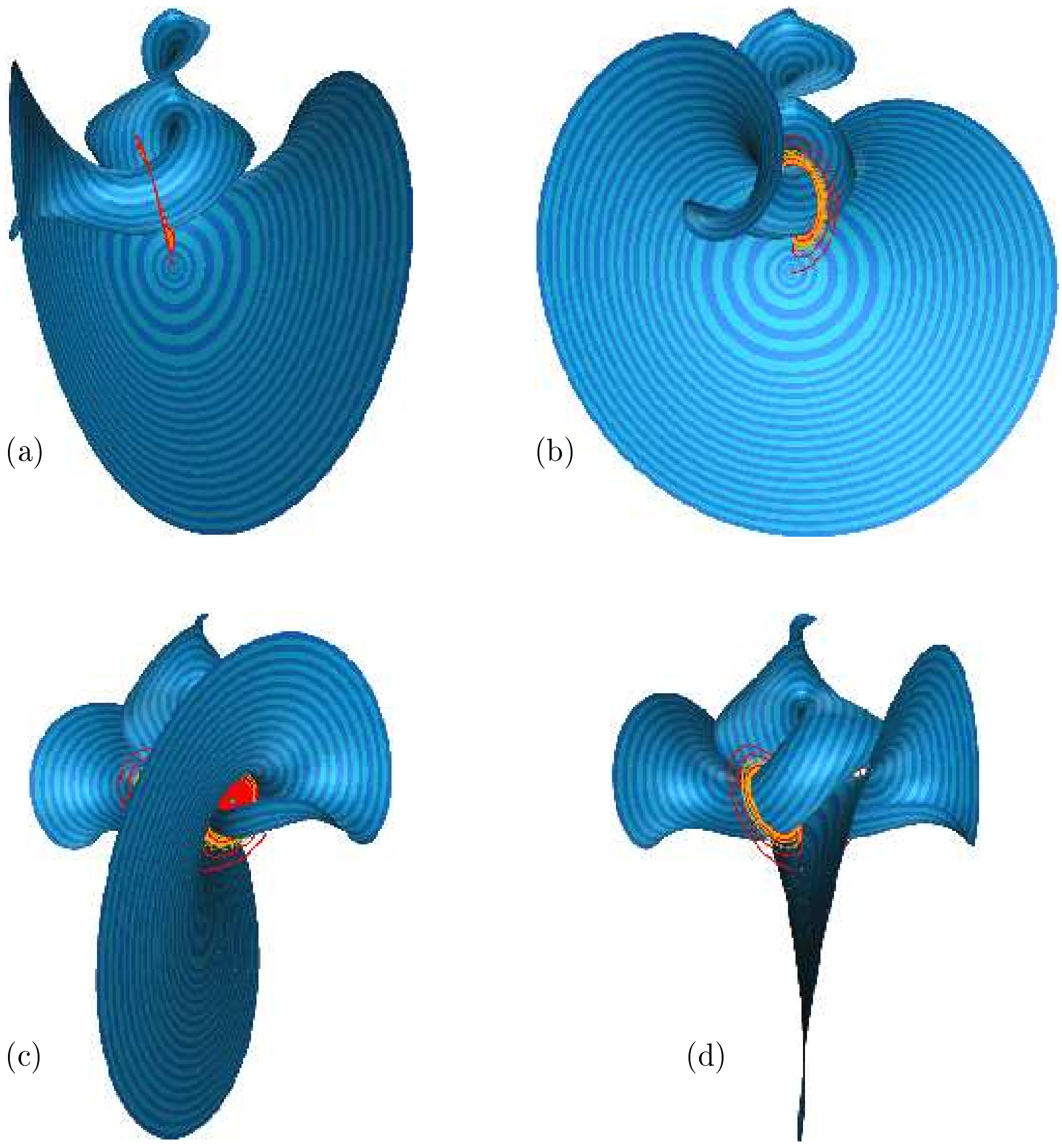


Figure 4: *The stable manifold* $W^s(0)$ *of the Lorenz system viewed along the* y*-axis (a), with rotations about the* z*–axis (vertical axis) of 40 degrees (b), 120 degrees (c), and 140 degrees (d), respectively.*

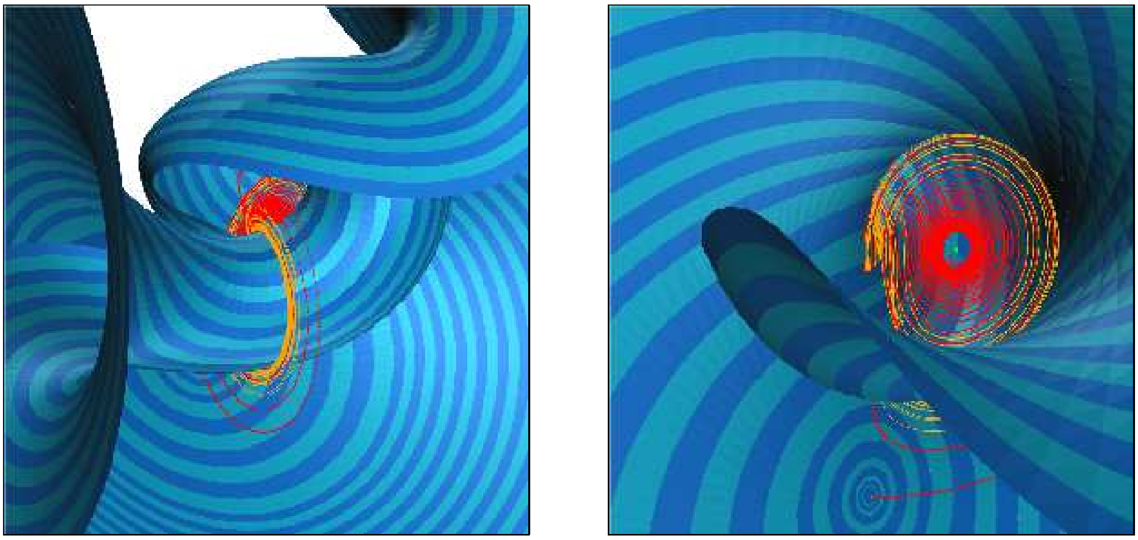


Figure 5: *Two close–up views of the stable manifold $W^s(0)$ of the Lorenz system.*

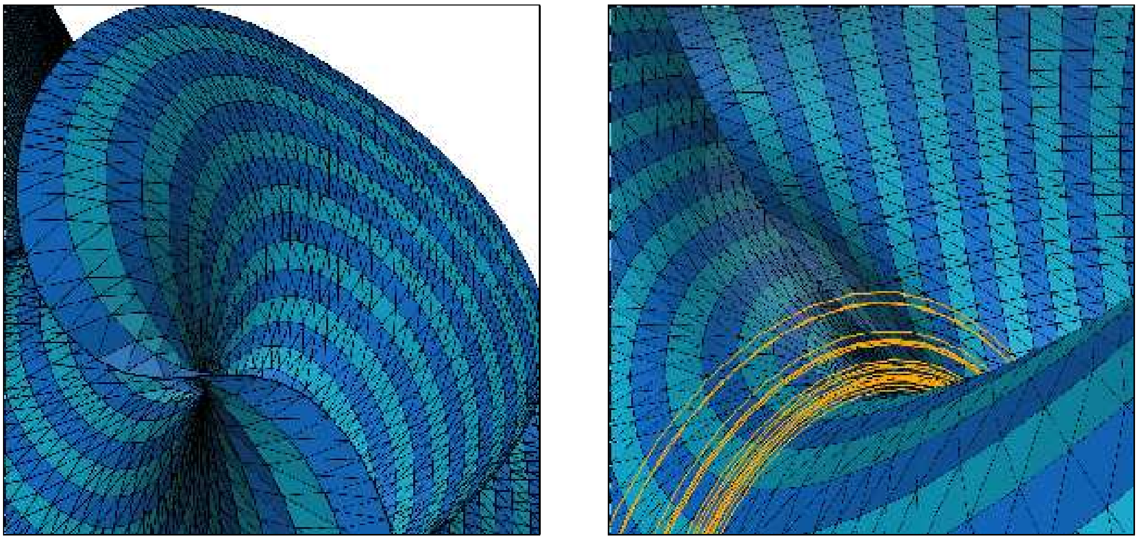
4 Examples

Our algorithm does not depend on the dynamics on the manifold. For example, it is not influenced by the ratio of the eigenvalues of the linear part at the saddle point. This is demonstrated in Section 4.1, where two stable eigenvalues of the origin differ by a factor of order 10. Furthermore, the algorithm has no problems dealing with complex conjugated eigenvalues, which is illustrated in Section 4.2. There, we also show how the algorithm handles convergence to a limit cycle. All figures have been rendered with the package Geomview [15].

4.1 The Lorenz system

The well-known Lorenz system [11] is defined as

$$\left\{ \begin{array}{rcl} \dot{x} & = & \sigma(y-x) \\ \dot{y} & = & \varrho x - y - xz \\ \dot{z} & = & xy - \beta z. \end{array} \right. \tag{4}$$



(a) (b)

Figure 6: *Mesh points are added (a) and removed (b) in the course of the computation of the stable manifold* $W^s(0)$ *of the Lorenz system.*

The origin is always an equilibrium, and for the standard parameters $\sigma = 10$, $\varrho = 28$, and $\beta = 2\frac{2}{3}$ it has a stable manifold for which the corresponding eigenvalues are $\lambda^s = -\beta$ and $\lambda^{ss} \approx -22.8$. This stable manifold is known to have a complicated structure, because it spirals into the butterfly-shaped attractor without actually intersecting it. Note that the z-axis is invariant and part of the stable manifold.

We compute $W^s(0)$ using the following accuracy parameters. The vector field is integrated with a fixed time step Runge-Kutta order four integrator, taking a maximum of 150 steps. The time step is 0.001 initially and is adapted as points are integrated further away from the equilibrium such that always at least 2 integration steps are taken to determine the next circle. The distance between circles is at least $\Delta_{min} = 0.01$, and it is controlled by the accuracy conditions explained in [13] ($0.3 < \alpha < 0.2$, and $0.1 < \Delta\alpha < 1.0$). We start with an initial circle of 20 points at distance $\delta = 1.0$ from the origin. The distance between leaves is kept at a maximum of $\Delta_{\mathcal{F}} = 2.0$, and is at least $\delta_{\mathcal{F}} = 0.5$. The final circle contains 885 points.

The result of the computation is shown in Figures 4, 5, and 6. The stable manifold $W^s(0)$ is shown in blue, equilibria are green, and the attractor is yellow. For completeness, the one-dimensional unstable manifold $W^u(0)$ is shown in red (computed with the package DsTool [2]). Figure 4 shows four views of the entire manifold computed up to about arclength 100. The two close-up views of $W^s(0)$ in Figure 5 show the complicated interaction of the manifold and the attractor. Obviously, many leaves are added during this computation, but also some leaves are removed as they come too close to neighboring leaves. Both adaptations are illustrated in Figure 6. Animations that show how the algorithm grows the manifold can be found at `http://www.cds.caltech.edu/~hinke/lorenz/`.

4.2 Arneodo's system

The following example is taken from [1], and we call it Arneodo's system:

$$\begin{cases} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= \alpha\, x - x^2 - \beta\, y - z, \end{cases} \tag{5}$$

where $\alpha = 2.5$ and $\beta = 2$. This system has two equilibria, one at the origin and one at $(\alpha, 0, 0)$. For this choice of parameters, both equilibria are of saddle type. We concentrate on the equilibrium $(\alpha, 0, 0)$ with eigenvalues $\lambda^s \approx -1.15$ and $\lambda^u \approx 0.075 \pm 1.47i$, and consider its two-dimensional unstable manifold. This manifold is compact and accumulates on a limit cycle.

As in Section 4.1, we use a fixed time step order four Runge Kutta integrator, with a maximum of 1000 steps. Due to the spiralling behavior on the manifold, the next circle will typically be much more than 1 integration step away, so we take the time step 0.01 and keep it fixed during the entire computation. Further accuracy parameters are $\Delta_{\mathcal{F}} = 0.1$, $\delta_{\mathcal{F}} = 0.025$, $\Delta_{min} = 0.01$, $0.2 < \alpha < 0.3$, and $0.001 < \Delta\alpha < 0.1$. We start with 40 points on an initial circle of radius $\delta = 0.1$. The algorithm first grows circles uniformly until the manifold gets close to the limit cycle; see Figure 7 (a). As there is slow radially outward growth near the limit cycle, it becomes harder to find points at distance Δ from the boundary. When this happens, the algorithm allows for a variable growth rate per leaf, such that the convergence process can be completed; see Figure 7 (b). An animation of the convergence process can be found at `http://www.cds.caltech.edu/~hinke/arneodo/`.

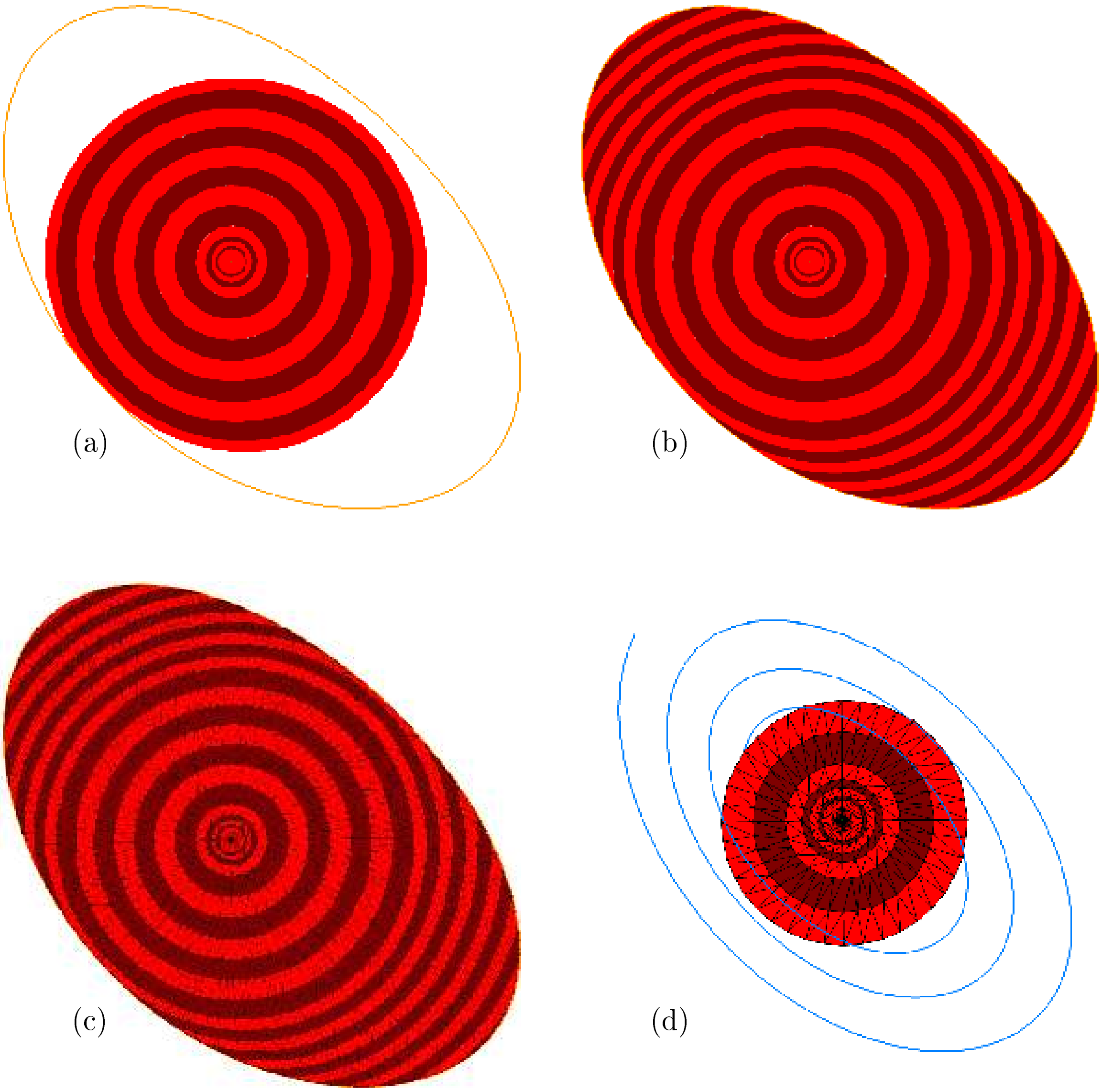


Figure 7: *The unstable manifold* $W^u(\alpha, 0, 0)$ *of Arneodo's system grows with the same speed in each leave until it hits a limit cycle (a). Then the convergence process starts until the entire manifolds has been computed (b), resulting in the mesh shown in (c). That the vector field does not always point outward with respect to the last circle is illustrated by the orbit in blue (d).*

The resulting mesh is shown in Figure 7 (c). We assume convergence in a leaf when the Δ-step in this leaf cannot be taken larger than 10^{-3}. Note that the algorithm is not affected by the spiralling behavior of orbits on the manifold. Even if the vector field points radially inward at certain points on the boundary of the manifold, the next circle can be properly found; see Figure 7 (d).

5 Discussion

We presented an algorithm for globalizing two-dimensional stable and unstable manifolds of equilibria in three-dimensional vector fields. This algorithm computes the manifold as a set of (topological) circles. To ensure a prescribed mesh quality we find new points in a set of planes that are as orthogonal as possible to the last circle. This is done by solving a boundary value problem. In the course of the computation planes are added and removed and possible convergence to an attractor is detected.

Only a simple change in starting data is needed to compute the unstable manifold of a periodic orbit Γ. At each point r on Γ there is a well-defined unstable linear direction $v^u(r)$. By choosing a mesh on Γ, we define an initial circle as the set of points at distance δ from the mesh points r on Γ in the direction of $v^u(r)$. Growing this initial circle is now done in exactly the same way as for a saddle point; compare [12].

Our method is independent of the dynamics on $W^u(x_0)$ without a need for rescaling the vector field. Also, we do not require that the vector field points outwards with respect to the last circle, as is required in [8, 17]. When adding new mesh points, that is, new planes, interpolation is only used within the prescribed mesh distance, so that the interpolation error is always under control; compare [10]. The mesh is adapted to the local curvature of $W^u(x_0)$, so that the number of mesh points is minimal without sacrificing the accuracy. The speed of the computation is largely determined by the time needed for integration, which depends on the dynamics. Using a boundary solver means a more time consuming computation, but we find that the algorithm is quite fast.

There are two main sources of errors in the computation. First, there is the initial error that one makes by picking starting data on $E^u(x_0)$ at a prescribed distance δ from x_0. This error can be reduced by taking δ very small, but then it may take a very long time for orbits to grow sufficiently long. However, it should be noted that finding a better, that is, higher order approximation of $W^u_{\rm loc}(x_0)$ generally takes a long computation as well.

Second, there is the interpolation error between mesh points. Along rays of the mesh we monitor the curvature by the method in [9, 13] to ensure that mesh points along rays are not too far apart. This means that we use many points where the manifold locally folds sharply, but only few mesh points in slightly curved parts. However, along circles of the mesh the accuracy is controlled by the distance between the mesh points only and local curvature is not taken into consideration. A better accuracy, without using too many mesh points, could be obtained by monitoring the curvature along circles of the mesh as well. Note that we use simple linear interpolation between mesh points. Interpolating splines might also increase the accuracy in most cases.

Our algorithm can be generalized to compute m-dimensional unstable manifolds in n-dimensional phase spaces. The key idea is again to grow the manifold as a one-parameter family of $(m-1)$ spheres as the intersection with suitable linear subspaces orthogonal to the last sphere. In this way a prescribed mesh quality can be achieved. Finding a new point in such a subspace again amounts to solving a boundary value problem. Clearly, there are serious issues of data management and visualization. A detailed discussion of the general case is beyond the scope of this paper and will appear elsewhere.

Acknowledgments

This work was initiated during the author's participation in the Special Year on Emerging Applications of Dynamical Systems 1997/98 at the Institute for Mathematics and its Applications, Minneapolis. We thank W.-J. Beyn, M. Dellnitz, E.J. Doedel, J. Guckenheimer, and M.E. Johnson for helpful discussions and comments, and the IMA for its hospitality and support. B.K. thanks the California Institute of Technology and H.O. the University of Bristol for its hospitality.

References

[1] A. Arneodo, P. Coullet, E. Spiegel, and C. Tresser. Asymptotic chaos. *Physica D* **14**(3) (1985), 327–347.

[2] A. Back, J. Guckenheimer, M.R. Myers, F.J. Wicklin, and P.A. Worfolk. DsTool: Computer assisted exploration of dynamical systems. *Notices Amer. Math. Soc.* **39**(4) (1992), 303–309.

[3] M. Dellnitz and A. Hohmann. The computation of unstable manifolds using subdivision and continuation. in *Nonlinear Dynamical Systems and Chaos* (H.W. Broer, S.A. van Gils, I. Hoveijn, and F. Takens eds.), *PNLDE* **19** (Birkhäuser, 1996), 449–459.

[4] M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Num. Math.* **75** (1997), 293–317.

[5] E.J. Doedel, A.R. Champneys, T.R. Fairgrieve, Yu.A. Kuznetsov, B. Sandstede, and X.J. Wang. AUTO97 continnuation and bifurcation software for ordinary differential equations. Available via `ftp://ftp.cs.concordia.ca/pub/doedel/auto/` (1997).

[6] E.J. Doedel. Private communications at the IMA (October 1997).

[7] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, (Springer, 1986).

[8] J. Guckenheimer and P. Worfolk. Dynamical systems: Some computational problems. in *Bifurcations and Periodic Orbits of Vector Fields* (D. Schlomiuk ed.), (Kluwer Academic Publishers, 1993), 241–277.

[9] D. Hobson. An efficient method for computing invariant manifolds of planar maps. *J. Comput. Phys.* **104**(1) (1993), 14–22.

[10] M.E. Johnson, M.S. Jolly, and I.G. Kevrekidis. Two-dimensional invariant manifolds and global bifurcations: some approximation and visualization studies. *Numerical Algorithms* **14** (1997), 125–140.

[11] E.N. Lorenz. Deterministic nonperiodic flows. *J. Atmospheric Sci.* **20** (1963), 130–141.

[12] B. Krauskopf and H.M. Osinga. Globalizing two-dimensional unstable manifolds of maps. *Int. J. Bif. Chaos* **8**(3) (1998), 483–503.

[13] B. Krauskopf and H.M. Osinga. Growing 1D and quasi 2D unstable manifolds of maps. *J. Comp. Phys.* **146**(1) (1998), 404–419.

[14] J. Palis and W. de Melo. *Geometric Theory of Dynamical Systems*, (Springer, 1982).

[15] M. Phillips, S. Levy, and T. Munzner. Geomview: An Interactive Geometry Viewer. *Notices of the Amer. Math. Soc.* **40** (1993), 985–988. This software and the accompanying manual are available at `http://www.geom.umn.edu/`.

[16] S. Wiggins. *Chaotic Transport in Dynamical Systems*, (Springer, 1992).

[17] P. Worfolk. Private communications (May 1997).