SIAM J. APPLIED DYNAMICAL SYSTEMS
Vol. 3, No. 3, pp. 232–260

A Fast Method for Approximating Invariant Manifolds*

John Guckenheimer† and Alexander Vladimirsky†

Abstract. The task of constructing higher-dimensional invariant manifolds for dynamical systems can be computationally expensive. We demonstrate that this problem can be locally reduced to solving a system of quasi-linear PDEs, which can be efficiently solved in an Eulerian framework. We construct a fast numerical method for solving the resulting system of discretized nonlinear equations. The efficiency stems from decoupling the system and ordering the computations to take advantage of the direction of information flow. We illustrate our approach by constructing two-dimensional invariant manifolds of hyperbolic equilibria in $\boldsymbol{R}^3$ and $\boldsymbol{R}^4$.

Key words. global stable and unstable manifolds, numerical method, ordered upwind

AMS subject classifications. 37D10, 37M20, 35F30, 65N22

DOI. 10.1137/030600179

1. Introduction. Invariant manifolds are important in many application areas. In the context of dynamical systems theory, stable and unstable manifolds are fundamental geometric structures. They partition phase-spaces into sets of points with the same forward and backward limit sets. We cite the following three ways in which problems involving stable and unstable manifolds of equilibria arise:

1. Stable and unstable manifolds play a role in global bifurcation. Homoclinic and heteroclinic bifurcations occur at nontransverse intersections of stable and unstable manifolds. For example, a particular global bifurcation of the Kuramoto–Sivashinsky equation is examined in [19] using the method for approximating invariant manifolds developed in [18].
2. In studying the structure of weak shock waves for hyperbolic systems of conservation laws, the admissibility of a traveling-wave ansatz depends on the existence of a heteroclinic orbit connecting the left-state/right-state equilibria; see, for example, [35]. Such an orbit exists if the stable manifold of u_r intersects the unstable manifold of u_l.
3. For systems with multiple attractors whose basins cover all but the set of measure zero, a basin boundary can often be obtained from the stable manifolds of equilibria with a single unstable direction. Such delineation of basins is an important practical task. For example, transient stability analysis of power systems deals with the stability properties after an *event disturbance* modeled as a time-localized (*fault-on*) change in the vector field. The key test is to determine if a fault-on trajectory ends up inside the desired stability region of the postfault system [2]. On the other hand, in designing

*Received by the editors April 28, 2003; accepted for publication (in revised form) by M. Golubitsky February 10, 2004; published electronically July 6, 2004.
http://www.siam.org/journals/siads/3-3/60017.html

†Department of Mathematics, Cornell University, Ithaca, NY 14853 (gucken@cam.cornell.edu , vlad@math.cornell.edu). The first author was partially supported by the National Science Foundation and Department of Energy. The second author was partially supported by a National Science Foundation Postdoctoral Research Fellowship.

hierarchical controls, a high-bandwidth part of the control-structure might be turned off once the system reaches a desired basin of attraction [1].

This paper presents a fast numerical method for approximating stable and unstable manifolds of equilibrium points of a vector field. Given a smooth vector field $\boldsymbol{f}$ in $\boldsymbol{R}^n$ and a hyperbolic saddle point $\boldsymbol{y_0}$, the corresponding invariant manifolds are defined as

$$W^s(\boldsymbol{y_0}) = \left\{ \boldsymbol{y} \in \boldsymbol{R}^n \mid \lim_{t\to+\infty} \phi_f^t(\boldsymbol{y}) = \boldsymbol{y_0} \right\},$$
$$W^u(\boldsymbol{y_0}) = \left\{ \boldsymbol{y} \in \boldsymbol{R}^n \mid \lim_{t\to-\infty} \phi_f^t(\boldsymbol{y}) = \boldsymbol{y_0} \right\},$$

where ϕ_f^t is the time flow of the vector field $\boldsymbol{f}$. In the vicinity of $\boldsymbol{y_0}$, the original dynamical system

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}) \tag{1}$$

is well approximated by its linearization

$$\boldsymbol{y}' = D\boldsymbol{f}(\boldsymbol{y_0})\boldsymbol{y}. \tag{2}$$

Moreover, by the stable manifold theorem [14], the invariant manifolds of $\boldsymbol{y_0}$ are tangent to the corresponding manifolds for the linearized system (2), i.e., tangent to the respective stable ($E^s(\boldsymbol{y_0})$) and unstable ($E^u(\boldsymbol{y_0})$) eigenspaces of the matrix $D\boldsymbol{f}(\boldsymbol{y_0})$.

If the invariant manifold $W^u(\boldsymbol{y_0})$ is only one-dimensional, its approximation can be easily obtained by choosing an initial point in the unstable subspace of (2) and by integrating forward in time.[1] However, for higher-dimensional cases, the manifold consists of an infinite number of trajectories, making the task of approximating the manifold much more challenging. This problem has attracted a considerable amount of attention. In section 2 we discuss several previously available approximation methods.

We note that dynamical systems with multiple time-scales present an additional degree of complexity: for such systems, obtaining a "geometrically satisfactory" representation of the manifold is often very expensive computationally. Indeed, the most natural idea (i.e., to approximate the manifold by following a finite number of trajectories in W^u for some fixed time T) will not work very well in this case. For a simple example, consider the linear vector field

$$\boldsymbol{y}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{y} \tag{3}$$

with a saddle point at the origin and $W^u(0)$ coinciding with the x-y plane.

Observation 1.1. In Figure 1 we show some typical trajectories in this plane and the images of a small circle around the origin under the flow ϕ_f^t. The following two well-known problems with this approach will be even more pronounced for the nonlinear case:

[1] In the rest of the paper we will mainly refer to the problem of constructing the unstable invariant manifold. The stable manifold can be constructed similarly by a time reversal (i.e., by considering the vector field $-\boldsymbol{f}$).

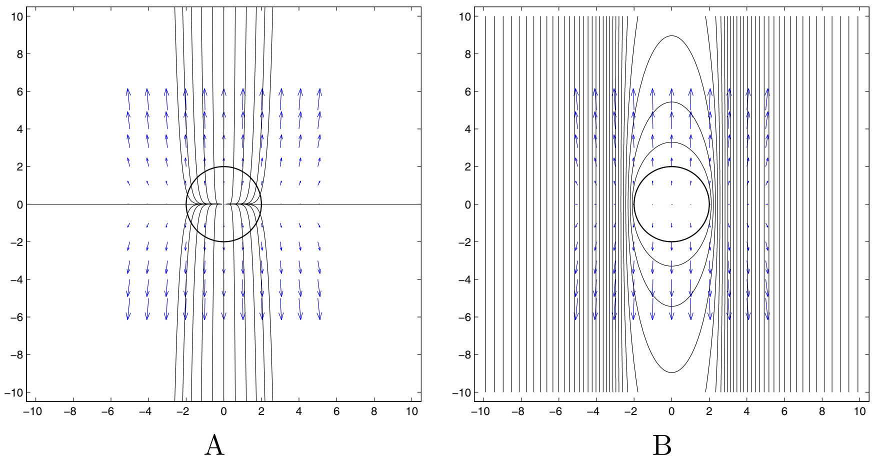


Figure 1. *A simple linear multiple-time-scale example. Trajectories are chosen to be equidistributed on a circle of radius* $R_{init} = 2$.

- If one starts with a number of points equidistributed on a small circle around $\boldsymbol{y_0}$ in $E^u(\boldsymbol{y_0})$, the speed of those points varies significantly even for relatively tame problems.
- The respective trajectories (Figure 1A) and the ϕ_f^t-images of the initial circle (Figure 1B) do not provide for a good mesh-representation of the manifold object.

The latter reflects an inherent conflict between the objectives of respecting the flow direction (which, after all, defines the manifold) and stressing the geometric properties of the manifold in a mesh-representation.

Our approach reconciles the two objectives: To approximate an invariant manifold of co-dimension k, we formulate a system of k quasi-linear PDEs satisfied by the manifold's local parameterization (section 3); we then solve that system locally in an Eulerian framework (section 4), thus limiting possible distortions of the mesh due to a variance of speeds for different directions inside the manifold.

The resulting discretized equations are solved very efficiently by decoupling them and ordering the computation of simplex-patches (added to the earlier computed manifold representation) to take advantage of the direction of information flow (section 6). Our algorithm can be viewed as an extension of the ordered upwind methods (OUMs) introduced in [32, 33] for static Hamilton–Jacobi PDEs. These methods solve a boundary value problem in $O(N \log N)$ operations, where N is the total number of mesh-points (section 5). The resulting triangulated mesh approximates a compact subset

$$\mathrm{W}^u_{\Sigma}(\boldsymbol{y_0}) = \{\boldsymbol{y} \in W^u(\boldsymbol{y_0}) \mid \sigma(\boldsymbol{y}) \leq \Sigma\},$$

where $\sigma(\boldsymbol{y})$ is defined as a distance-along-the-trajectory from $\boldsymbol{y}$ to $\boldsymbol{y_0}$, and Σ is a prespecified stopping criterion. For a desired mesh-scale Δ, the compactness of $\mathrm{W}^u_{\Sigma}(\boldsymbol{y_0})$ and the non-degeneracy of simplex-patches ensure that $N = O(\Delta^{-k})$.

In section 7 we use our method to construct a two-dimensional stable manifold of the origin for the Lorenz system. In section 8 we consider a dynamical system describing two pendula coupled by a torsional spring and compute two-dimensional unstable and stable manifolds for one of its saddle equilibria. Though the rate of convergence of the method is not proven, in section 9 we provide numerical evidence to confirm the first-order global accuracy. This is consistent with the local truncation error of order $O(\Delta^2)$ analytically derived in section 4.1. We conclude by discussing the limitations of our approach and possible future extensions in section 10.

2. Prior methods.

A number of previously available techniques for computing invariant manifolds all follow the same general principle, which is that an invariant k-dimensional manifold is grown as a one-parameter family $\{M_i\}$ of topological $(k-1)$-spheres, where M_0 is taken to be a small $(k-1)$-sphere around $\boldsymbol{y_0}$ in $E^u(\boldsymbol{y_0})$. However, the resulting methods are quite diverse as a result of different choices for

- the family-parameter of $\{M_i\}$ (e.g., integration time, distance along the trajectory to $\boldsymbol{y_0}$, geodesic distance to $\boldsymbol{y_0}$, etc.),
- data structures used to store the M_i's, and
- the algorithm for producing M_{i+1} given M_i.

The simplest implementation of this idea was illustrated in section 1 as follows: M_0 is approximated by a finite number of equidistant markers, the family-parameter is chosen to be the integration time, and M_{i+1} is approximated by the position of markers approximating M_i after some time Δt. As shown in Figure 1A, initially equidistributed markers quickly converge and/or drift apart due to the geometric stiffness[2] of the vector field $\boldsymbol{f}$; thus, an additional step of redistributing markers along M_{i+1} is required. Moreover, since the size of the M_i's varies, additional markers might be needed to ensure the quality of approximation (e.g., the maximum distance Δx between adjacent markers in M_{i+1}). As a result, an accurate approximation will require that small Δt be used even if marker-trajectories are computed with infinite precision.

Another problem with this approach is a highly nonuniform distance between M_i and M_{i+1} (see Figure 1B), resulting in a poor geometric approximation of the manifold even if each $(k-1)$-sphere in the family is known perfectly. A method for alleviating this difficulty was introduced by Johnson, Jolly, and Kevrekidis in [18]. Their method uses a rescaling of the vector field ("reparameterizing to integrate with respect to arclength in space-time"), which ensures the same speed along all the trajectories; i.e., the family-parameter becomes a distance-from-$\boldsymbol{y_0}$-along-the-trajectory. Unfortunately, the produced mesh still need not be the best geometric representation of the manifold since the local distance between M_i and M_{i+1} is now determined by the ratio of "M_i-normal" and "M_i-tangential" components of the rescaled vector field. We note that the computationally expensive marker-redistribution is still required at each step due to the geometric stiffness (since the rescaling of the vector field does not change the trajectories).

A method ensuring the constant distance between M_i and M_{i+1} was introduced by Guckenheimer and Worfolk in [15]. The family-parameter is chosen to be the geodesic distance from $\boldsymbol{y_0}$. The markers on M_i are moved with a unit speed for a time Δt in the direction outwards-normal to M_i within the locally determined tangent k-plane. The approximation resulting for M_{i+1} might still require marker addition/redistribution, but only due to the different size

[2] Here and throughout the paper, by *geometric stiffness* we mean the highly nonuniform rates of separation for nearby trajectories on different parts of the manifold.

of M_{i+1} and not due to the geometric stiffness. A tangent k-plane is locally determined for each marker in M_i using the adjacent markers and the direction of the vector field. Thus, this procedure becomes very sensitive wherever $\boldsymbol{f}$ is nearly tangential to M_i, leading to excessively expensive restrictions for the ratio of $\Delta t/\Delta x$.

Another method also using the geodesic distance as a family-parameter was introduced by Krauskopf and Osinga in [20, 21, 22]. For a given marker $\boldsymbol{y} \in M_i$, a locally normal $(n-k+1)$-plane $\mathcal{F}_{\boldsymbol{y}}$ is determined using the adjacent nodes in M_i. Then a shooting method is used to solve the following boundary value problem: Find a point (not necessarily a marker!) $\boldsymbol{z} \in M_i$ such that its trajectory intersects $\mathcal{F}_{\boldsymbol{y}}$ at some point $\tilde{\boldsymbol{z}}$ and $\|\tilde{\boldsymbol{z}} - \boldsymbol{y}\| = \Delta t$. A collection of $\tilde{\boldsymbol{z}}$'s is used as an approximation of M_{i+1}; as before, some new markers might be required due to the bigger size of M_{i+1}. An explicit bound on the overall computational error is available, and the quality of the resulting mesh is ensured by adding/removing the markers on M_i, depending on the manifold's local geometry [21]. We note that the above procedure is robust even if the vector field $\boldsymbol{f}$ is locally tangential to M_i, but the shooting method becomes much more computationally expensive in that case. In addition, solving the boundary value problem for each marker becomes even more challenging for $k > 2$ because the search space for $\boldsymbol{z}$ is no longer one-dimensional.

Remark 2.1. All of the above methods are explicit in the sense that only the representation of M_i is used to produce M_{i+1} and the order of computation of the markers on M_{i+1} is unimportant. Thus, these methods' computational complexity is generally proportional to the total number (across all of the M_i's) of used mesh-points N. However, N will depend not only on the required accuracy in manifold-approximation but also on the choice of family-parameter. Moreover, the proportionality constants involved can be quite large, depending on k (e.g., for the marker-redistribution) and on the orientation of $\boldsymbol{f}$ relative to the M_i's.

Several other numerical techniques are not based on growing a family of M_i's.

A method introduced by Doedel [11] uses a single computed trajectory in $W^u(\boldsymbol{y_0})$ as an input for the boundary value solver of AUTO [10] to perform continuation in the ray-angle parameter. The manifold is approximated between $M_{init} = M_0$ and M_{final} by a sequence of trajectories $\{\boldsymbol{z^j}\}$ such that $\boldsymbol{z^j}(0) \in M_0$, $\boldsymbol{z^j}(\tau_j) \in M_{final}$, and $\|\boldsymbol{z^j}(\tau_j) - \boldsymbol{z^{j+1}}(\tau_{j+1})\| = \Delta$. Starting with the initial trajectory $\boldsymbol{z^0}$, the collocation methods are repeatedly used to produce $\boldsymbol{z^{j+1}}$ based on $\boldsymbol{z^j}$. If P_j is a hyperplane transversal to $\boldsymbol{f}$ at $\boldsymbol{z^j}(\tau_j)$, then $\boldsymbol{z^{j+1}}$ is sought with one end point on M_0 and the other lying on P_j distance Δ away from $\boldsymbol{z^j}(\tau_j)$. The resulting sequence $\{\boldsymbol{z^j}\}$ is well spaced near M_{final} but may not be uniformly spaced near M_0.

A new method by Henderson [16] is based on integrating an individual trajectory together with a second-order approximation to the manifold along that trajectory. The surface is constructed as a collection of k-dimensional strips centered at such trajectories; the use of these (nonintersecting) strips provides uniform bounds on the spacing of the trajectories. The implementation heavily relies on the efficient data structures developed earlier for approximating implicitly defined manifolds [17]. This method is the first to directly model the curvature information in the direction transversal to the trajectories.

An algorithm introduced by Dellnitz and Hohmann in [6, 7] uses subdivision and cell-mapping-continuation techniques to produce an n-dimensional covering of the k-dimensional unstable manifold. A simplified version of this algorithm can be summarized as follows. The computational domain Q is subdivided into a number of small, nonintersecting n-dimensional "boxes." The initial covering is determined as a collection of those boxes covering $W^u_{loc}(\boldsymbol{y_0})$—a small neighborhood of $\boldsymbol{y_0}$ in $W^u(\boldsymbol{y_0})$. The iteratively repeated continuation stage adds new boxes to the collection if they are intersected by a $\phi_f^{\Delta t}$-image of some box(es) already in the collection. The process stops when no more boxes within Q can be added. The use of efficient (hierarchical) data structures allows storing only those boxes actually needed for the covering. The covering's growth reflects the anisotropy due to multiple time-scales present in the system; i.e., the more strongly unstable directions are covered first. The accuracy of the approximation depends upon the size of boxes in the resulting covering and upon the level of refinement of the initial covering (the relative sizes of the boxes compared to $W^u_{loc}(\boldsymbol{y_0})$); hence the algorithm can be quite memory intensive and may converge relatively slowly, especially for $k = n-1$. The efficiency of this algorithm also strongly depends on the contraction transversal to the manifold: weaker contraction will require a much finer initial covering—otherwise, the cell mapping will produce a coarse n-dimensional covering of W^u.

We note that the method in [6, 7] is currently the only one implemented for $k > 2$. Several other methods briefly described above were formulated for the general case, but, to the best of our knowledge, the current implementations rely on $k = 2$.

3. PDE approach to manifold-approximation. In contrast to the methods discussed in the previous section, we compute an invariant manifold as a collection of adjacent k-dimensional simplex-patches. The $(k-1)$-dimensional boundary of the current collection is used to attach new tentative simplexes, whose exact position in $\boldsymbol{R}^n$ is computed using a PDE for the local parameterization of that manifold.

We begin by considering a relatively simple case of a two-dimensional manifold in $\boldsymbol{R}^3$. If $(x_1, x_2, g(x_1, x_2)) = (\boldsymbol{x}, g(\boldsymbol{x})) = \boldsymbol{y}$ is a local parameterization of an invariant manifold, then the vector field evaluated on it should be tangential to the graph of $g(x_1, x_2)$, i.e.,

$$\boldsymbol{f}\,(x_1, x_2, g(x_1, x_2)) \begin{bmatrix} \frac{\partial}{\partial x_1} g(x_1, x_2) \\ \frac{\partial}{\partial x_2} g(x_1, x_2) \\ -1 \end{bmatrix} = 0, \tag{4}$$

should hold wherever this parameterization is valid. Our general method can be outlined as follows:

- The above first-order quasi-linear PDE can be solved to "continue" the manifold since the boundary condition for g is specified on the last previously computed manifold "boundary."
- The initial "boundary" is approximated by a discretized small circle around $\boldsymbol{y_0}$ in $E^u(\boldsymbol{y_0})$.
- Once a new triangle-patch of the manifold is computed and *Accepted*, the computational "boundary" (*AcceptedFront*) is modified to include it, new tentative (or *Considered*) patches are added to the computational domain, and the PDE is solved on them using the new (local) coordinates.

This process is discussed in detail in section 6 and illustrated in section 7. Here, we simply note that, unlike a general quasi-linear PDE, equation (4) always has a smooth solution as long as the chosen parameterization remains valid. Thus, switching to local coordinates when solving the PDE allows us to avoid checking the continued validity of the parameterization.

The above derivation can be repeated to obtain a single PDE defining an invariant manifold of codimension 1 in $\boldsymbol{R}^n$: the number of equations corresponds to the number of linearly independent vectors orthogonal to the manifold's tangent space, i.e., to the manifold's codimension.

In this spirit, we now consider the general problem of constructing a k-dimensional invariant manifold of a vector field $\boldsymbol{f} : \boldsymbol{R}^n \to \boldsymbol{R}^n$. Switching to a suitable coordinate system, we assume that the manifold's local parameterization is $(x_1, \dots, x_k, g_1(x_1, \dots, x_k), \dots, g_{n-k}(x_1, \dots, x_k)) = (\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{y} \in \boldsymbol{R}^n$, where $\boldsymbol{x} \in \boldsymbol{R}^k$ and $\boldsymbol{g} : \boldsymbol{R}^k \mapsto \boldsymbol{R}^{n-k}$. As in the codimension 1 case, the PDE is derived from the condition that the vector field evaluated on the manifold should lie in its tangent space. Therefore, for every $j \in \{1, \dots, (n-k)\}$,

$$\sum_{i=1}^{k} \frac{\partial g_j}{\partial x_i}(x_1, \dots, x_k) f_i(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})) = f_{j+k}(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})) \tag{5}$$

should locally hold as long as the above parameterization is valid. This coupled system of $(n-k)$ quasi-linear PDEs can again be used to "continue" the manifold since the boundary conditions for the g_j's are specified on the last-previously-computed-manifold-"boundary." In this case, the initial "boundary" is approximated by a discretized $(k-1)$-sphere around $\boldsymbol{y_0}$ in $E^u(\boldsymbol{y_0})$ and the manifold grows as new k-dimensional simplexes are *Accepted*. The construction of manifolds of codimension 2 is illustrated in section 8.

Remark 3.1 (a historical note). The PDE approach for characterizing invariant surfaces goes back to at least the 1960s. In particular, the existence and smoothness of solutions for equations equivalent to (5) are the subjects of Sacker's analytical perturbation theory [30, 25]. Previous numerical techniques based on this formulation included time-marching finite difference schemes in "special coordinates" [9], iterative methods [8] based on a discrete version of Fenichel's graph transform [13], collocation methods [12], and spectral methods [24]. However, all this work was done for invariant tori computations, resulting in the following two very important distinctions from our method:

1. These prior methods assume the existence of a coordinate system in which the invariant torus is indeed globally a graph of the function. Such a coordinate system may be defined explicitly [9] or implicitly [8]. In the latter case it can be defined using normal/tangent bundles of a (previously constructed) invariant torus of a slightly perturbed vector field. This implies availability of a global mesh on which the PDE can be solved.
   For invariant manifolds of hyperbolic equilibria, such a mesh is not available a priori and has to be constructed in the process of "growing" the manifold (section 6.5).
2. For the invariant tori computations, the solution function $\boldsymbol{g}$ has periodic boundary conditions; hence, the discretized equations are inherently coupled and have to be solved simultaneously.
   For approximation of $W^u(\boldsymbol{y_0})$, all characteristics of the PDE start at the initial boundary (chosen in $E^u(\boldsymbol{y_0})$) and run "outward." Knowledge of the direction of information flow can be used to decouple the discretized system, resulting in a much faster computational method (section 5).

4. Eulerian discretization. Not surprisingly, the characteristics of (4) are exactly the (projections of the) trajectories of the original vector field. Thus, any attempt to solve it in the Lagrangian framework (i.e., by the method of characteristics or ray shooting) would bring us back to all the problems discussed in section 1. On the other hand, it was demonstrated in [33] that the discretized (semi-Lagrangian and Eulerian) equations resulting from certain nonlinear first-order PDEs can be solved very efficiently. This was our motivation for locally recasting this problem in a fully Eulerian framework.

For a two-dimensional invariant manifold in $\boldsymbol{R}^3$ (as formulated in (4)), let $G(x_1, x_2)$ be a piecewise-linear numerical approximation of the solution $g(x_1, x_2)$. Consider a simplex $\boldsymbol{y}\boldsymbol{y^1}\boldsymbol{y^2}$, where $\boldsymbol{y^i} = (x_1^i, x_2^i, G(x_1^i, x_2^i)) = (\boldsymbol{x^i}, G(\boldsymbol{x^i}))$ and $\boldsymbol{y} = (x_1, x_2, G(x_1, x_2)) = (\boldsymbol{x}, G(\boldsymbol{x}))$. We assume that the vertices $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$ are two adjacent mesh-points on the $AcceptedFront$ (the discretization of the current manifold "boundary"). Thus, $G(\boldsymbol{x^i})$'s are known and can be used in computing $G(\boldsymbol{x})$. Define the unit vectors $\boldsymbol{P_i} = (\boldsymbol{x} - \boldsymbol{x^i})/\|\boldsymbol{x} - \boldsymbol{x^i}\|$ and let P be a matrix with $\boldsymbol{P_i}$'s as its rows. This square matrix is invertible since $\boldsymbol{x}$ is chosen some distance away from the $AcceptedFront$. We note that a directional derivative for G in the direction $\boldsymbol{P_i}$ can be computed as

$$v_i(\boldsymbol{x}) = \frac{G(\boldsymbol{x}) - G(\boldsymbol{x^i})}{\|\boldsymbol{x} - \boldsymbol{x^i}\|}. \tag{6}$$

Therefore, if $\boldsymbol{v}$ is a column vector of v_i's, then $\nabla g(\boldsymbol{x}) \approx \nabla G(\boldsymbol{x}) = P^{-1}\boldsymbol{v}$, yielding the discretized version of (4):

$$\left[P^{-1}\boldsymbol{v}(\boldsymbol{x})\right]_1 f_1(\boldsymbol{x}, G(\boldsymbol{x})) \;+\; \left[P^{-1}\boldsymbol{v}(\boldsymbol{x})\right]_2 f_2(\boldsymbol{x}, G(\boldsymbol{x})) \;=\; f_3(\boldsymbol{x}, G(\boldsymbol{x})). \tag{7}$$

This nonlinear equation can be solved for $G(\boldsymbol{x})$ by the Newton–Raphson method or any other robust zero-solver. In addition, it has an especially simple geometric interpretation if the local coordinates are chosen so that $G(\boldsymbol{x^1}) = G(\boldsymbol{x^2}) = 0$. Setting $\hat{\boldsymbol{y}} = (\boldsymbol{x}, 0)$, we reduce the problem to finding the correct "tilt" for a simplex $\boldsymbol{y}\boldsymbol{y^1}\boldsymbol{y^2}$. If $\boldsymbol{u}$ is a unit vector normal to $\hat{\boldsymbol{y}}\boldsymbol{y^1}\boldsymbol{y^2}$, then solving (7) is equivalent to finding a number $\alpha \in \boldsymbol{R}$ such that $\boldsymbol{f}(\hat{\boldsymbol{y}} + \alpha\boldsymbol{u})$ lies in the plane defined by $\boldsymbol{y^1}$, $\boldsymbol{y^2}$, and $\boldsymbol{y} = \hat{\boldsymbol{y}} + \alpha\boldsymbol{u}$. (See Figure 2.)

This geometric interpretation can be extended to the general case[3] of a k-dimensional invariant manifold in $\boldsymbol{R}^n$. In this case, the $AcceptedFront$ is a $(k-1)$-dimensional mesh discretizing the currently computed manifold "boundary" and we consider a k-dimensional simplex $\boldsymbol{y}\boldsymbol{y^1}\ldots\boldsymbol{y^k}$, where $\boldsymbol{y^1}, \ldots, \boldsymbol{y^k} \in \boldsymbol{R}^n$ form a $(k-1)$-dimensional simplex in $AcceptedFront$ and $\boldsymbol{y}$ is a $Considered$ point near it. A local parameterization $\boldsymbol{g}(\boldsymbol{x})$ satisfying system (5) is numerically approximated by $\boldsymbol{G}(\boldsymbol{x})$; i.e., we assume $\boldsymbol{y^i} = (\boldsymbol{x^i}, \boldsymbol{G}(\boldsymbol{x^i}))$ and $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{G}(\boldsymbol{x}))$, where $\boldsymbol{x}, \boldsymbol{x^i} \in \boldsymbol{R}^k$ and $\boldsymbol{G} : \boldsymbol{R}^k \to \boldsymbol{R}^{n-k}$. We choose the parameterization so that $\boldsymbol{G}(\boldsymbol{x^i}) = \boldsymbol{0}$ and let $\hat{\boldsymbol{y}} = (\boldsymbol{x}, \boldsymbol{0})$. Let $\{\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{(n-k)}\}$ form an orthonormal basis for the orthogonal complement of the k-plane $\hat{\boldsymbol{y}}\boldsymbol{y^1}\ldots\boldsymbol{y^k}$. Then the task of linearly approximating system (5) is equivalent to finding the real numbers $\alpha_1, \ldots, \alpha_{(n-k)}$ such that, for $\boldsymbol{y} = \hat{\boldsymbol{y}} + \sum_{i=1}^{n-k} \alpha_i \boldsymbol{u}^i$, the vector $\boldsymbol{f}(\boldsymbol{y})$ lies in the k-plane defined by $\boldsymbol{y}, \boldsymbol{y^1}, \ldots, \boldsymbol{y^k}$.

[3] Of course, the explicit discretization formula is also available. We omit it here for the sake of brevity and notational clarity.

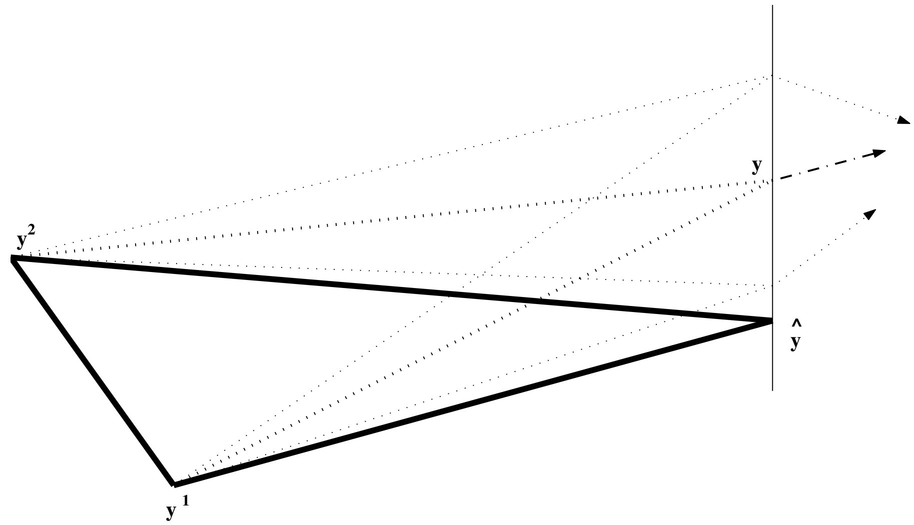


Figure 2. *Geometric interpretation of (7). The one-dimensional search-space corresponds to the manifold's codimension.*

The described discretization procedure is similar in spirit to an *implicit Euler method* for solving initial value problems since $\boldsymbol{y^i}$'s are assumed to be known and the vector field is computed at a yet-to-be-determined point $\boldsymbol{y}$.

4.1. Local truncation error and upwinding condition. Let L be the Lipschitz constant of $\boldsymbol{f}$ and let ν be the upper bound of $\|\nabla g\|$ on a Δ-neighborhood of $\boldsymbol{x}$. For $k = 2$, suppose that $\boldsymbol{y^1} = (\boldsymbol{x^1}, G(\boldsymbol{x^1}))$ and $\boldsymbol{y^2} = (\boldsymbol{x^2}, G(\boldsymbol{x^2}))$ lie on the manifold and that $\boldsymbol{y} = (\boldsymbol{x}, G(\boldsymbol{x}))$ solves (7). This means that $\boldsymbol{f}(\boldsymbol{y})$ lies in the plane of $\boldsymbol{y}\boldsymbol{y^1}\boldsymbol{y^2}$ and can be expressed as a linear combination

$$\boldsymbol{f}(\boldsymbol{y}) = \beta_1(\boldsymbol{y} - \boldsymbol{y^1}) + \beta_2(\boldsymbol{y} - \boldsymbol{y^2}), \tag{8}$$

where the real coefficients β_1 and β_2 satisfy the equation

$$P^T \begin{bmatrix} \beta_1 \|\boldsymbol{x} - \boldsymbol{x^1}\| \\ \beta_2 \|\boldsymbol{x} - \boldsymbol{x^2}\| \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{y}) \\ f_2(\boldsymbol{y}) \end{bmatrix}. \tag{9}$$

If $\boldsymbol{f}(\boldsymbol{y})$ is not parallel to $\boldsymbol{y^1}\boldsymbol{y^2}$, then $\beta_1 + \beta_2 \neq 0$, and a linear approximation of $\boldsymbol{y}$'s trajectory (i.e., the straight line through $\boldsymbol{y}$ in the direction $\boldsymbol{f}(\boldsymbol{y})$) will intersect the line $\boldsymbol{y^1}\boldsymbol{y^2}$ at the point $\tilde{\boldsymbol{y}} = (\beta_1 \boldsymbol{y^1} + \beta_2 \boldsymbol{y^2})/(\beta_1 + \beta_2)$. We note that

$$\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\| = \frac{\|\boldsymbol{f}(\boldsymbol{y})\|}{|\beta_1 + \beta_2|}. \tag{10}$$

Since, away from equilibria, $\|\boldsymbol{f}(\boldsymbol{y})\|$ is bounded from below, (10) implies $|\beta_1 + \beta_2|^{-1} = O(\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|)$.

Using the above notation, we can rewrite (7) in the form

$$G(\boldsymbol{x}) = \frac{\beta_1}{\beta_1 + \beta_2} G(\boldsymbol{x^1}) + \frac{\beta_2}{\beta_1 + \beta_2} G(\boldsymbol{x^2}) + \frac{f_3(\boldsymbol{x}, G(\boldsymbol{x}))}{\beta_1 + \beta_2}. \tag{11}$$

For $\tilde{\boldsymbol{x}} = (\beta_1 \boldsymbol{x}^{\mathbf{1}} + \beta_2 \boldsymbol{x}^{\mathbf{2}})/(\beta_1 + \beta_2)$ we can now express

$$\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|^2 \;=\; \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2 + \left(\frac{f_3(\boldsymbol{x}, G(\boldsymbol{x}))}{\beta_1 + \beta_2}\right)^2 \;=\; \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2 + \left(\frac{f_3(\boldsymbol{y})\,\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|}{\|\boldsymbol{f}(\boldsymbol{y})\|}\right)^2.$$

Therefore,

$$\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|^2 \;=\; \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2 \frac{\|\boldsymbol{f}(\boldsymbol{y})\|^2}{\|\boldsymbol{f}(\boldsymbol{y})\|^2 - (f_3(\boldsymbol{y}))^2}.$$

Thus, $O(\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|) = O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|)$, provided $|f_3| \ll \|\boldsymbol{f}\|$. This condition can be satisfied by a suitable choice of the coordinate system, e.g., if the point $\boldsymbol{x}$ is chosen so that $f_3(\tilde{\boldsymbol{x}}) = 0$. In that case, the "correction term" $f_3(\boldsymbol{y})/(\beta_1 + \beta_2)$ in formula (11) is of the order $O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2)$. For the solution $g(\boldsymbol{x})$ of PDE (4),

$$\begin{aligned} g(\boldsymbol{x}) &= g(\tilde{\boldsymbol{x}}) + \nabla g(\boldsymbol{x}) \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2) \\ &= g(\tilde{\boldsymbol{x}}) + \nabla g(\boldsymbol{x}) \cdot \frac{\beta_1(\boldsymbol{x} - \boldsymbol{x}^{\mathbf{1}}) + \beta_2(\boldsymbol{x} - \boldsymbol{x}^{\mathbf{2}})}{\beta_1 + \beta_2} + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2). \end{aligned}$$

Combining the above with (8), we obtain

$$g(\boldsymbol{x}) = g(\tilde{\boldsymbol{x}}) + \frac{f_1(\boldsymbol{x}, G(\boldsymbol{x}))\frac{\partial}{\partial x_1}g(\boldsymbol{x}) + f_2(\boldsymbol{x}, G(\boldsymbol{x}))\frac{\partial}{\partial x_2}g(\boldsymbol{x})}{\beta_1 + \beta_2} + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2)$$

and

$$\begin{aligned} |g(\boldsymbol{x}) - G(\boldsymbol{x})| &\le \left| g(\tilde{\boldsymbol{x}}) + \frac{f_1(\boldsymbol{x}, G(\boldsymbol{x}))\frac{\partial}{\partial x_1}g(\boldsymbol{x}) + f_2(\boldsymbol{x}, G(\boldsymbol{x}))\frac{\partial}{\partial x_2}g(\boldsymbol{x})}{\beta_1 + \beta_2} - G(\boldsymbol{x}) \right| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2) \\ &\le \left| g(\tilde{\boldsymbol{x}}) + \frac{f_1(\boldsymbol{x}, g(\boldsymbol{x}))\frac{\partial}{\partial x_1}g(\boldsymbol{x}) + f_2(\boldsymbol{x}, g(\boldsymbol{x}))\frac{\partial}{\partial x_2}g(\boldsymbol{x})}{\beta_1 + \beta_2} - G(\boldsymbol{x}) \right| \\ &\quad + \frac{2L\nu}{|\beta_1 + \beta_2|}|g(\boldsymbol{x}) - G(\boldsymbol{x})| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2). \end{aligned}$$

Since g solves the PDE, we obtain

$$\begin{aligned} |g(\boldsymbol{x}) - G(\boldsymbol{x})| &\le \left| g(\tilde{\boldsymbol{x}}) + \frac{f_3(\boldsymbol{x}, g(\boldsymbol{x}))}{\beta_1 + \beta_2} - G(\boldsymbol{x}) \right| + \frac{2L\nu}{|\beta_1 + \beta_2|}|g(\boldsymbol{x}) - G(\boldsymbol{x})| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2) \\ &\le \left| g(\tilde{\boldsymbol{x}}) + \frac{f_3(\boldsymbol{x}, G(\boldsymbol{x}))}{\beta_1 + \beta_2} - G(\boldsymbol{x}) \right| + \frac{L(2\nu + 1)}{|\beta_1 + \beta_2|}|g(\boldsymbol{x}) - G(\boldsymbol{x})| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2). \end{aligned}$$

Setting $C = L(2\nu + 1)/\|\boldsymbol{f}(\boldsymbol{y})\|$ and recalling (10), (11) gives

$$\begin{aligned} (1 - C\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|)\;|g(\boldsymbol{x}) - G(\boldsymbol{x})| &\le \left| g(\tilde{\boldsymbol{x}}) + \frac{f_3(\boldsymbol{x}, G(\boldsymbol{x}))}{\beta_1 + \beta_2} - G(\boldsymbol{x}) \right| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2) \\ &= \left| g(\tilde{\boldsymbol{x}}) - \frac{\beta_1}{\beta_1 + \beta_2}G(\boldsymbol{x}^{\mathbf{1}}) - \frac{\beta_2}{\beta_1 + \beta_2}G(\boldsymbol{x}^{\mathbf{2}}) \right| + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2). \end{aligned}$$

Let $\boldsymbol{x^m} = (\boldsymbol{x^1} + \boldsymbol{x^2})/2$. Since $\tilde{\boldsymbol{x}}$ is on the line $\boldsymbol{x^1 x^2}$ and it was assumed that $G(\boldsymbol{x^i}) = g(\boldsymbol{x^i})$, the linear approximation yields

$$g(\tilde{\boldsymbol{x}}) = (\beta_1 G(\boldsymbol{x^1}) + \beta_2 G(\boldsymbol{x^2}))/(\beta_1 + \beta_2) + O(\|\tilde{\boldsymbol{x}} - \boldsymbol{x^m}\|^2) + O(\|\boldsymbol{x^1} - \boldsymbol{x^2}\|^2).$$

Thus,

$$|g(\boldsymbol{x}) - G(\boldsymbol{x})| \leq \frac{O\left(\|\tilde{\boldsymbol{x}} - \boldsymbol{x^m}\|^2\right) + O\left(\|\boldsymbol{x^1} - \boldsymbol{x^2}\|^2\right) + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2)}{(1 - C\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|)}. \tag{12}$$

If $\tilde{\boldsymbol{x}}$ lies in between $\boldsymbol{x^1}$ and $\boldsymbol{x^2}$, and if the triangle $\boldsymbol{x x^1 x^2}$ has sides of length $\leq \Delta$, then formula (12) yields a local truncation error of order $O(\Delta^2)$. Correspondingly, we expect the global approximation error of order $O(\Delta)$ for the entire mesh. The rigorous analysis of the global error is outside the scope of this paper, but in section 9 we provide numerical evidence to confirm the first-order accuracy.

The requirement that $\tilde{\boldsymbol{x}}$ should lie in between $\boldsymbol{x^1}$ and $\boldsymbol{x^2}$ ensures that interpolation (rather than extrapolation) is used for $G(\tilde{\boldsymbol{x}})$. Moreover, it corresponds to the fundamental stability condition for solving first-order PDEs: the mathematical domain of dependence should be included in the numerical domain of dependence. For our problem this means that $G(\boldsymbol{x})$ should be computed using the correct triangle—the triangle through which the corresponding (approximate) trajectory runs. Thus, having computed $\boldsymbol{y} = (\boldsymbol{x}, G(\boldsymbol{x}))$ by (7) using two adjacent mesh-points $\boldsymbol{y^i}$ and $\boldsymbol{y^j}$, we need to verify the following additional *upwinding condition*: the linear approximation to $\boldsymbol{y}$'s trajectory should intersect the line $\boldsymbol{y^i y^j}$ at the point $\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{x}}, G(\tilde{\boldsymbol{x}}))$ lying between $\boldsymbol{y^i}$ and $\boldsymbol{y^j}$ (see Figure 3) or, equivalently, $\boldsymbol{f}(\boldsymbol{y})$ should point from the newly computed simplex $\boldsymbol{y y^i y^j}$. If the upwinding criterion is satisfied (i.e., $\beta_1, \beta_2 \geq 0$), formula (10) provides the length of the linear approximation of $\boldsymbol{y}$'s trajectory inside $\boldsymbol{y y^i y^j}$. Both the upwinding condition and formula (10) can be similarly extended for $k > 2$.

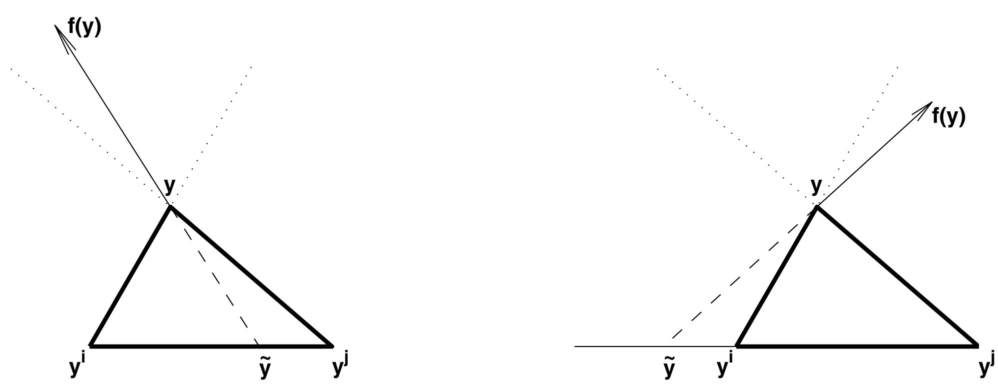


Figure 3. *Examples of acceptable (left) and unacceptable (right) approximations of $\boldsymbol{f}(\boldsymbol{y})$. The range of upwindings directions is shown by dotted lines; the local linear approximation to the trajectory is shown by dashed line; $\tilde{\boldsymbol{y}}$ is its intersection with the line $\boldsymbol{y^i y^j}$. In the second case the upwinding criterion is not satisfied and the update for $\boldsymbol{y}$ should be computed using another segment of AcceptedFront.*

5. Ordered upwind methods. OUMs were originally introduced by Sethian and Vladimirsky to solve a class of problems in anisotropic control theory and anisotropic front propagation described by static Hamilton–Jacobi–Bellman PDEs [32, 33]. A finite-difference discretization of a nonlinear boundary value problem normally leads to a system of N nonlinear coupled discretized equations, where N is the total number of mesh-points in the computational domain. The solution to that system is usually obtained iteratively, while each iteration involves recomputing the values at all of the mesh-points. Such iterative schemes can be quite slow even in conjunction with Gauss–Seidel relaxation techniques. OUMs provide an alternative by using the partial information about the direction of information flow to essentially decouple the system and to solve the equations one by one. Several extensions of these methods were introduced for hybrid control problems [34] and for phase-space multiple-arrivals computations [31].

The decoupling introduced in [32] hinges on the notion of "optimality" and the variational properties of the PDEs arising in the control-theoretic context. This formulation was heavily used in proving convergence to the viscosity solution of the Hamilton–Jacobi–Bellman PDE [33]. However, the more general idea behind the methods was to allow *space-marching* for the boundary value problems—not unlike explicit forward-time marching for initial-boundary value problems. In essence, the solution can be "marched" (on the mesh) from the boundary using the characteristic information, and a new (smaller) boundary value problem can be posed using the newly computed "boundary"—the current divide between the already-computed (*Accepted*) and not-yet-touched (Far) mesh-points. The mesh discretization of that "new boundary" is referred to as $AcceptedFront$; the not-yet-*Accepted* mesh-points, which are adjacent to the $AcceptedFront$, are designated $Considered$. A tentative value can be computed for each $Considered$ mesh-point $\boldsymbol{x}$ under the assumption that its characteristic intersects the $AcceptedFront$ in some vicinity of that mesh-point (designated $NF(\boldsymbol{x})$). All $Considered$ points are sorted based on the $SortValue$ (usually defined as the time-to-travel to $\boldsymbol{x}$ from the boundary along its characteristic). A typical step of the algorithm consists of choosing the $Considered$ $\bar{\boldsymbol{x}}$ with the smallest $SortValue$ and making it *Accepted*. This operation modifies the $AcceptedFront$ ($\bar{\boldsymbol{x}}$ in; other mesh-points possibly out), and causes a possible recomputation of all the not-yet-*Accepted* mesh-points near $\bar{\boldsymbol{x}}$.

This "space-marching" is based on the principle of "local" solution reconstruction from characteristics and on some notion of an entropy-like condition (i.e., no characteristics emerging from shocks). Both of these are applicable for a much wider class of first-order PDEs. In [31] OUMs were successfully used to treat the linear Liouville PDE. The applicability of OUMs to general quasi-linear first-order PDEs is still an open question [36]. However, the particular computational problem considered in this paper has an additional simplifying property: the PDEs (4) and (5) are solved only locally, and hence the solution remains smooth at every point. On the other hand, unlike in the previous OUMs, the mesh is not known in advance and is built in the process of computation. In adding a tentative simplex-patch (with a $Considered$ vertex) we attempt to provide for "good" geometric properties of the mesh (e.g., simplex aspect ratio) and to ensure that the parameterization is locally well-conditioned (e.g., $\|\nabla g\|$ should be small on that simplex). The vector field near $AcceptedFront$ determines the order in which the correct "tilts" for tentative simplex-patches are computed and the $Considered$ mesh-points are *Accepted*. This ordering has the effect of reducing the approximation error (a mesh-point $\boldsymbol{y}$ first computed from a relatively far part of $NF(\boldsymbol{y})$ is likely to be recomputed before it gets *Accepted*). Below we outline the general structure of the algorithm and provide a detailed description of individual components in section 6. As in the original OUMs, the computational complexity of the algorithm is $O(N \log N)$, where the $(\log N)$ factor results from the necessity of maintaining a sorted list of *Considered* mesh-points.

Ordered upwind method for building invariant manifolds.

1. Use the linearization $(E^u(\boldsymbol{y_0}))$ to initialize *AcceptedFront* and one "layer" of *Considered*s.
2. Evaluate the tentative coordinates for *Considered*s.
3. Find the *Considered* mesh-point $\bar{\boldsymbol{y}}$ which is the closest (in the sense of trajectory distances) to *AcceptedFront*.
4. Move $\bar{\boldsymbol{y}}$ to *Accepted* and update the *AcceptedFront*.
5. Remove/add the *Considered* mesh-points to reflect changes to *AcceptedFront*.
6. Recompute the coordinates for all the *Considered* $\boldsymbol{y}$ such that $\bar{\boldsymbol{y}} \in NF(\boldsymbol{y})$.
7. If *Considered* is not empty (and "stopping criteria" are not met) then go to 3.

6. Implementation details.

The current implementation is specifically geared toward two-dimensional manifolds in $\boldsymbol{R}^n$, even though a generalization of most of the following is straightforward (except for section 6.3 and parts of section 6.5 which explicitly rely on $k = 2$). Our goal is to construct a simplicial complex approximating the manifold with the preferred triangle side of length Δ (but definitely less than 2Δ). Throughout this section we call a mesh-triangle s *adjacent* to $\boldsymbol{y}$ if $\boldsymbol{y}$ is one of the vertices of s; we also refer to mesh-points $\boldsymbol{y^i}$ and $\boldsymbol{y^j}$ as adjacent (or connected) if both of them are vertices of the same triangle.

6.1. Initialization.

The algorithm is initialized using the linearization of the vector field. $E^u(\boldsymbol{y_0})$ is determined as a span of eigenvectors of $D\boldsymbol{f}(\boldsymbol{y_0})$ corresponding to the eigenvalues with positive real part.

Any simple closed curve around $\boldsymbol{y_0}$ in $E^u(\boldsymbol{y_0})$ can be used as an initial boundary I, provided that curve is transversal to the linearized vector field. In the examples considered in the following sections, I was chosen to be a circle of radius R_{init} centered at $\boldsymbol{y_0}$. More generally, the transversality condition can always be satisfied by choosing an ellipse corresponding to the relevant eigenvectors.

The initial boundary I is then approximated using N_0 *Accepted* mesh-points so that the distance between all adjacent mesh-points $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$ is at most Δ. For each such adjacent pair, the segment $\boldsymbol{y^1 y^2}$ is placed onto the *AcceptedFront* and an equilateral triangle is constructed with the vertices at $\boldsymbol{y^1}$, $\boldsymbol{y^2}$, and $\hat{\boldsymbol{y}}$, where the new *Considered* mesh-point $\hat{\boldsymbol{y}}$ lies in $E^u(\boldsymbol{y_0})$ and $\|\boldsymbol{y_0} - \hat{\boldsymbol{y}}\| = R_{init} + \Delta\sqrt{3}/2$. (See Figure 4.)

6.2. Computing coordinates for *Considered*s.

Once the correct "tilt" is computed, each *Considered* mesh point $\boldsymbol{y}$ is a vertex of a triangle with the other vertices $\boldsymbol{y^1}$, $\boldsymbol{y^2}$ on the *AcceptedFront*. Initially, however, that triangle is built to be locally tangential to the manifold; i.e., only $\hat{\boldsymbol{y}}$ is known at first instead of $\boldsymbol{y}$ (see Figure 2). If $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$ are on I, then $\hat{\boldsymbol{y}}$ is chosen in $E^u(\boldsymbol{y_0})$, as described above; otherwise, the position of $\hat{\boldsymbol{y}}$ is selected in the plane of the previously *Accepted* triangle adjacent to $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$ (as described in section 6.5). Given $\hat{\boldsymbol{y}}$, the discretized version of the PDE(s) is solved to obtain the "normal component(s)" of $\boldsymbol{y}$.

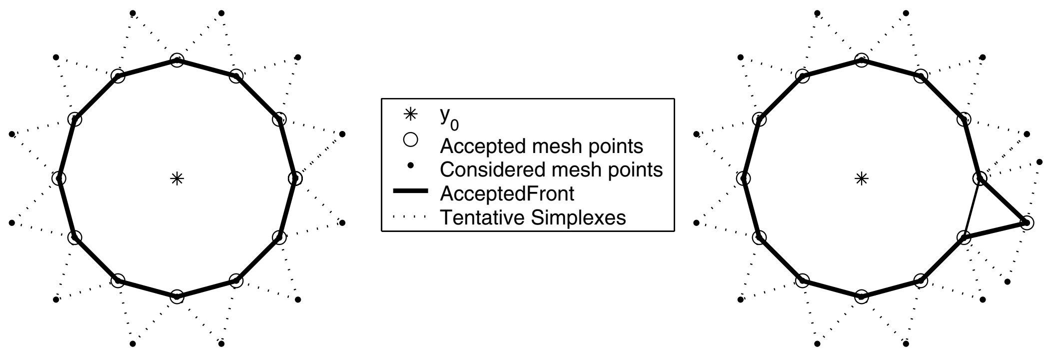


Figure 4. *Initialization of $AcceptedFront$ in $E^u(\boldsymbol{y_0})$ (left) and changes to $AcceptedFront$ once the first of $Considered$ points is Accepted (right). The newly added/accepted triangle generally does not lie in $E^u(\boldsymbol{y_0})$; once Accepted, it defines the plane in which the new tentative triangles are initially chosen.*

As described in section 4, the newly computed $\boldsymbol{y}$ is a valid $Considered$ point if it satisfies the "upwinding condition." If that condition is not satisfied, we recompute $\boldsymbol{y}$ using other segments on $AcceptedFront$ near $\hat{\boldsymbol{y}}$. In general, given two adjacent mesh-points $\boldsymbol{y^i}$ and $\boldsymbol{y^j}$ on $AcceptedFront$, we can form a "virtual simplex" $\hat{\boldsymbol{y}}\boldsymbol{y^i}\boldsymbol{y^j}$ which is then used to compute the value for $\boldsymbol{y}$ even if it is not directly adjacent to $\boldsymbol{y^i}$ or $\boldsymbol{y^j}$. A $NearFront$ $NF(\hat{\boldsymbol{y}})$ is defined as a collection of segments on $AcceptedFront$ within the distance R_{NF} from $\hat{\boldsymbol{y}}$.

$NF(\hat{\boldsymbol{y}})$ is used to restrict the set of virtual simplexes, which will be potentially checked to find the one satisfying the upwinding criterion—the vector $\boldsymbol{f}(\boldsymbol{y})$ should be pointing out of the simplex used to compute $\boldsymbol{y}$ (Figure 5 illustrates this for a simplified case $\hat{\boldsymbol{y}} = \boldsymbol{y}$).

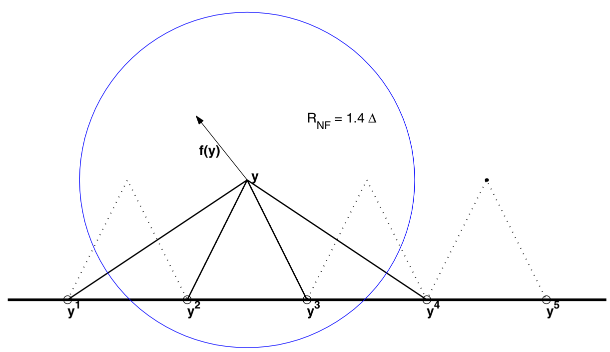


Figure 5. *Use of $NF(\hat{\boldsymbol{y}})$ to build virtual simplexes for evaluating $\boldsymbol{y}$. The first evaluation is performed using $\boldsymbol{y^2}$ and $\boldsymbol{y^3}$ as the Accepted mesh-points adjacent to $\hat{\boldsymbol{y}}$; the resulting approximation for $\boldsymbol{f}(\boldsymbol{y})$ shows that the upwinding condition is not satisfied and that a virtual simplex using $\boldsymbol{y^3}\boldsymbol{y^4}$ should be considered next.*

Remark 6.1. Generally, one needs to select the value for R_{NF} based on the behavior of the vector field near $AcceptedFront$—if the locally tangential components dominate the locally normal components, it will not be possible to satisfy the upwinding criterion unless R_{NF} is sufficiently large. (Note that this will also increase the local truncation error of section 4.1 by a factor of R_{NF}.)

On the other hand, this truly becomes a problem only if this local-tangentiality holds everywhere along the $AcceptedFront$: the fact that $Considered$ mesh-points are ordered based on $SortValue$ allows for a subsequent recomputation of coordinates of $\boldsymbol{y}$ once the position of $AcceptedFront$ changes. (See Figure 6.)

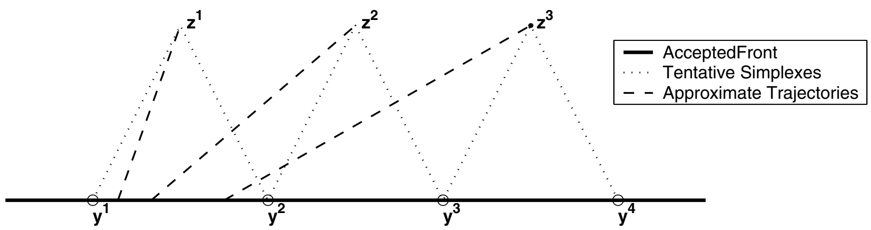


Figure 6. *Ordering acceptance of Considered based on $SortValue$ decreases the computational stencil; this reduces both the minimum sufficient R_{NF} and the local truncation error. Even though all three $\boldsymbol{z^i}$'s can be computed using $\boldsymbol{y^1 y^2}$, this is not really necessary. Once $\boldsymbol{z^1}$ becomes Accepted, both $\boldsymbol{z^2}$ and $\boldsymbol{z^3}$ can be computed from $\boldsymbol{z^1 y^2}$; once $\boldsymbol{z^2}$ becomes Accepted, $\boldsymbol{z^3}$ can be recomputed using $\boldsymbol{z^2 y^3}$. Thus, in this example, valid coordinate updates will be eventually computed even if $R_{NF} = \Delta$.*

6.3. Relaxing the upwinding condition.

The $AcceptedFront$ is a one-dimensional object, and the very first evaluation of $\boldsymbol{y}$'s coordinates will indicate the correct search-direction to satisfy the upwinding condition (Figure 5).

Unfortunately, $\boldsymbol{f}(\boldsymbol{y})$ gives only an approximation of the direction of information flow since the "normal" components (i.e., $\boldsymbol{y} - \hat{\boldsymbol{y}}$) are computed numerically from the first-order accurate discretization of the PDE. As a result, when the adjacent segments in $NF(\hat{\boldsymbol{y}})$ do not lie on the same line, it is possible that the upwinding condition will not be satisfied by any virtual simplex. (See Figure 7).

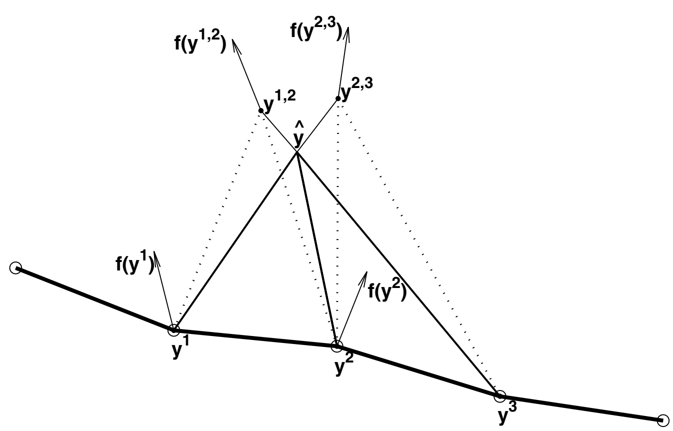


Figure 7. *Deadlock situation: according to $\boldsymbol{f}(\boldsymbol{y^{1,2}})$, $\boldsymbol{y}$'s trajectory should intersect $\boldsymbol{y^2 y^3}$; according to $\boldsymbol{f}(\boldsymbol{y^{2,3}})$, the trajectory should intersect $\boldsymbol{y^1 y^2}$. Thus, neither virtual simplex satisfies the strict upwinding condition and the relaxation of the criteria is needed. The directions of $\boldsymbol{f}(\boldsymbol{y^1})$ and $\boldsymbol{f}(\boldsymbol{y^2})$ show that the update $\boldsymbol{y^{1,2}}$ satisfies the relaxed criterion.*

Remark 6.2. Thus, we use the following relaxation of the upwinding criterion: Let $\hat{\boldsymbol{y}}$ be a $Considered$ point and let $\boldsymbol{y^{i,j}}$ be its new coordinates computed from a virtual simplex $s = \hat{\boldsymbol{y}}\boldsymbol{y^i}\boldsymbol{y^j}$. The *relaxed upwinding condition* is satisfied if there exists a point $\tilde{\boldsymbol{y}}$ on an $AcceptedFront$ segment $\boldsymbol{y^i}\boldsymbol{y^j}$ such that the projection of $\boldsymbol{f}(\tilde{\boldsymbol{y}})$ onto s is collinear with $\tilde{\boldsymbol{y}}\hat{\boldsymbol{y}}$. In practice, an alternative version of this condition is easier to verify: it suffices to check that the projections of $\boldsymbol{f}(\boldsymbol{y^i})$ and $\boldsymbol{f}(\boldsymbol{y^j})$ onto the plane of s lie outside of that simplex.

Remark 6.3. As formulated in section 4.1, the upwinding criterion is implicit: it cannot be verified until the tentative value $\boldsymbol{y^{i,j}}$ is computed. In contrast, the above *relaxed upwinding criterion* is explicit in nature since it concerns only the directions of the vector field on $\boldsymbol{y^i}\boldsymbol{y^j}$.

We note that, even when an explicit (relaxed) upwinding criterion is used, the "tilt" $(\boldsymbol{y} - \hat{\boldsymbol{y}})$ is still computed from the implicit formula (7). Moreover, in such cases we still have $\|\tilde{\boldsymbol{y}} - \boldsymbol{y}\| = O(R_{NF}\Delta)$ and the local truncation error derived in section 4.1 is still valid.

The above reasoning clearly uses the fact that $k = 2$. In general, the $AcceptedFront$ will be a $(k-1)$-dimensional object and both the search in $NF(\hat{\boldsymbol{y}})$ and the upwinding-relaxation procedures will have to be more complicated.

In our implementation, the relaxed upwinding is only used to deal with the deadlocks at $Considered$ points tagged as $Lagging$ (immediately adjacent to more than two segments of $AcceptedFront$ yet possessing no valid update). In all other cases, relaxation is postponed since subsequent modifications to $AcceptedFront$ may allow for the strict upwinding condition to be satisfied. As a result, the relaxation is applied very infrequently (e.g., at 24 mesh-points out of 77,500 in the example considered in section 8).

6.4. Computing the $SortValue$ and sorting $Considered$s.

Our decoupling orders the acceptance of $Considered$ points based on their "distance-along-the-trajectory-to-$\boldsymbol{y_0}$." That distance $\sigma(\boldsymbol{y})$ can be estimated as a sum of the "distance-along-the-trajectory-to-$AcceptedFront$" and $\sigma(\tilde{\boldsymbol{y}})$, where $\tilde{\boldsymbol{y}}$ is the intersection of $\boldsymbol{y}$'s trajectory with a segment $\boldsymbol{y^i}\boldsymbol{y^j}$ on the $AcceptedFront$ (see Figure 3).

Once the new coordinates for a $Considered$ point $\boldsymbol{y}$ are computed and the upwinding criterion is satisfied, we obtain a linear approximation to $\boldsymbol{y}$'s trajectory and can estimate its $SortValue$: $\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|$ can be computed by formula (10) and $\sigma(\tilde{\boldsymbol{y}})$ can be approximated by linearly interpolating $\sigma(\cdot)$ on $\boldsymbol{y^i}\boldsymbol{y^j}$. Since $\boldsymbol{y}$ is computed from $\boldsymbol{y^i}\boldsymbol{y^j}$, we recall that $\boldsymbol{f}(\boldsymbol{y}) = \beta_1(\boldsymbol{y} - \boldsymbol{y^i}) + \beta_2(\boldsymbol{y} - \boldsymbol{y^j})$ for some $\beta_1, \beta_2 \geq 0$ and

$$SortValue(\boldsymbol{y}) = \sigma(\boldsymbol{y}) \;\approx\; \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\| + \sigma(\tilde{\boldsymbol{y}}) \;\approx\; \frac{\|\boldsymbol{f}(\boldsymbol{y})\| + \beta_1\sigma(\boldsymbol{y^i}) + \beta_2\sigma(\boldsymbol{y^j})}{\beta_1 + \beta_2}. \tag{13}$$

If the (relaxed) upwinding criterion for $\boldsymbol{y}$ cannot be satisfied by any segment in $NF(\boldsymbol{y})$, then we leave $\boldsymbol{y} = \hat{\boldsymbol{y}}$ and assume $SortValue(\boldsymbol{y}) = +\infty$. Such a $Considered$ point will never get $Accepted$ unless the upwinding criterion is later satisfied in the subsequent recomputations (triggered by changes to $AcceptedFront$ near $\boldsymbol{y}$).

We use a heap-sort data structure to maintain the sorting of the $Considered$ points based on their $SortValue$s. As a result, selecting $\bar{\boldsymbol{y}}$ (stage 3 of the algorithm) can be performed in $O(1)$ operations, but every time a $Considered$ point's $SortValue$ changes, its position in the heap-sort should change as well. This resorting can be performed in $O(\log K)$ operations, where $K < N$ is the current number of $Considered$ mesh-points.

Remark 6.4. As noted in section 5, the general OUMs require using $T(\boldsymbol{y})$, the time-to-travel-along-the-characteristic, as a $SortValue$. However, this choice is dictated by the necessity of building the "correct" global (weak) solution to the PDE—any other ordering will risk running through the shocks and/or violating the entropy conditions [36]. In our current work, the characteristics are the trajectories of a smooth vector field and the solution is computed only locally; hence shocks cannot occur if Δ is sufficiently small. This smoothness of the solution enables us to use different sorting criteria, including $\sigma(\boldsymbol{y})$ (as above), $d(\boldsymbol{y})$ (the geodesic distance-to-$\boldsymbol{y_0}$), and the geodesic (or along-the-trajectory) distance to $AcceptedFront$. Our particular choice ($SortValue = \sigma(\boldsymbol{y})$) is a result of an empirical trade-off: it requires a lesser R_{NF} than ($SortValue = d(\boldsymbol{y})$) and generally decreases the length of $AcceptedFront$ as compared to ($SortValue = T(\boldsymbol{y})$).

6.5. Changing $AcceptedFront$ and extending the mesh.

Every $Considered$ point $\boldsymbol{y}$ is a vertex of at least one *tentative triangle* $\boldsymbol{y}\boldsymbol{y^1}\boldsymbol{y^2}$, where $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$ are adjacent mesh-points on the $AcceptedFront$. As a $Considered$ mesh-point $\bar{\boldsymbol{y}}$ becomes $Accepted$, this tentative triangle becomes *fully accepted* and the $AcceptedFront$ has to be modified accordingly. (Note that the newly added triangle will always use the segment $\boldsymbol{y^1}\boldsymbol{y^2}$ adjacent to $\bar{\boldsymbol{y}}$—even if that point was computed using some other segment $\boldsymbol{y^i}\boldsymbol{y^j}$ in $NF(\bar{\boldsymbol{y}})$.) The changes to $AcceptedFront$ proceed in two stages as follows:

1. Removal from $AcceptedFront$ of each segment $\boldsymbol{y^i}\boldsymbol{y^j}$ shared by two fully accepted triangles (or used by just one such triangle if both $\boldsymbol{y^i}$ and $\boldsymbol{y^j}$ are on the initial boundary I).
2. Adding to $AcceptedFront$ segments $\bar{\boldsymbol{y}}\boldsymbol{y^j}$, for all $AcceptedFront$ mesh-points $\boldsymbol{y^j}$ adjacent to $\bar{\boldsymbol{y}}$.

Once the $AcceptedFront$ has been modified, it may be necessary to extend the mesh near $\bar{\boldsymbol{y}}$. The existing mesh includes $Accepted$ points and a narrow band of $Considered$ points near $AcceptedFront$. If two mesh-points $\boldsymbol{y^k}, \boldsymbol{y^l} \notin I$ are adjacent, then we will refer to $\boldsymbol{y^k}\boldsymbol{y^l}$ as a *perimeter segment* if that segment is used as an edge by a unique triangle. The $AcceptedFront$ plays the role of an approximate boundary for (locally) solving the PDE. Correspondingly, if $\boldsymbol{y}$ is $Accepted$ but not on $AcceptedFront$, then there should be no perimeter segments adjacent to $\boldsymbol{y}$. Moreover, if $\boldsymbol{y^k}\boldsymbol{y^l}$ is on $AcceptedFront$, then it should not be a perimeter segment either. Yet if one of these points was just $Accepted$, the segment may be on the perimeter of the existing mesh and some local mesh-building is required to create the second triangle adjacent to $\boldsymbol{y^k}\boldsymbol{y^l}$. The following heuristic algorithm is similar to the "advancing front mesh generation" method described in [27].

Local mesh-extension algorithm. Let $L = \|\boldsymbol{y^k} - \boldsymbol{y^l}\|$ be the length of a perimeter segment $\boldsymbol{y^k}\boldsymbol{y^l}$. Let γ be the smallest *outer* angle formed by this segment, i.e., $\gamma = \min(\angle \boldsymbol{y^j}\boldsymbol{y^k}\boldsymbol{y^l}, \angle \boldsymbol{y^k}\boldsymbol{y^l}\boldsymbol{y^m})$, where $\boldsymbol{y^j}\boldsymbol{y^k}$ and $\boldsymbol{y^l}\boldsymbol{y^m}$ are perimeter segments adjacent to $\boldsymbol{y^k}\boldsymbol{y^l}$. Also, since $\boldsymbol{y^k}, \boldsymbol{y^l} \notin I$, there already exists a unique fully accepted triangle s_{kl} with these two vertices.

A second triangle adjacent to $\boldsymbol{y^k}\boldsymbol{y^l}$ is created by one of the following three procedures:

1. If $\gamma \geq \pi$, then we introduce a new mesh-point $\hat{\boldsymbol{y}}$ (in the plane defined by s_{kl}) to form

an isosceles triangle with $\boldsymbol{y^k y^l}$ as its base and sides $\boldsymbol{y^k \hat{y}}$, $\boldsymbol{y^l \hat{y}}$ of length L_1, where

$$L_1 = \begin{cases} 2\,L & \text{if } \ L \le \frac{\Delta}{2}; \\ \Delta & \text{if } \ \frac{\Delta}{2} < L \le \frac{\Delta}{0.55}; \\ 0.55\,L & \text{if } \ \frac{\Delta}{0.55} < L. \end{cases}$$

(See Figure 8A.) The constants used above are purely heuristic and are intended to balance our preference for nearly regular triangles against the desired mesh-scale Δ; see [27] for further details.

2. If $\angle \boldsymbol{y^j y^k y^l} = \gamma$ is acute, then a triangle $\boldsymbol{y^j y^k y^l}$ is added to the mesh. If $\|\boldsymbol{y^j y^l}\| > 2\Delta$, then that triangle is split in two by adding a new *Considered* mesh-point $\boldsymbol{\hat{y}}$. (See Figure 8B.)
3. If $\gamma \in [\frac{\pi}{2}, \pi)$, then we compute R_{jkl} and R_{klm}, the radii of circles passing through the respective triples of points. (The radius is assumed $+\infty$ if the corresponding outer angle is $\ge \pi$.) Without loss of generality, assume that $R_{jkl} \ge R_{klm}$. If $R_{klm} < L_1$, then a triangle $\boldsymbol{y^k y^l y^m}$ is added to the mesh without adding any new mesh-points. (See Figure 8C.) Otherwise, we create a new isosceles triangle $\boldsymbol{\hat{y} y^k y^l}$, as described above (see procedure 1).

We note that if a new mesh-point is created, then the choice of $\boldsymbol{\hat{y}}$ merely fixes the local coordinate system in which the "normal" components of $\boldsymbol{y}$ are next computed from $NF(\boldsymbol{y})$ (stage 6 of the algorithm) as described in section 6.2.

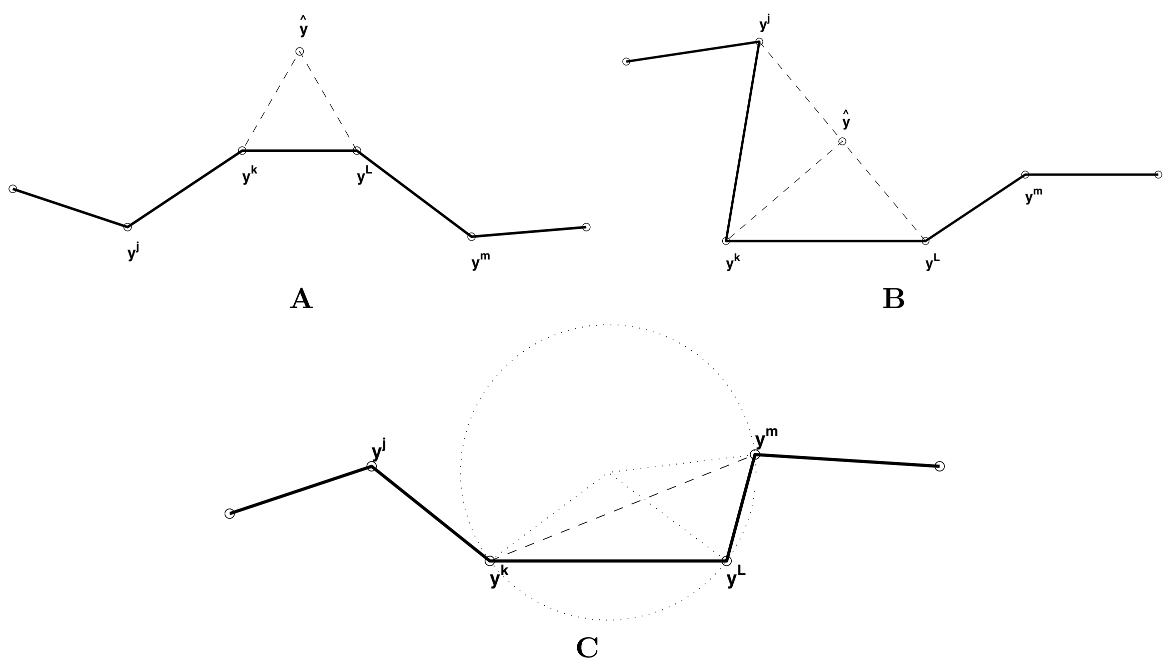


Figure 8. *Local mesh generation: three different procedures for extending the mesh at* $\boldsymbol{y^k y^l}$. *New segments are shown by dashed lines. If created, the new mesh-point is labeled* $\boldsymbol{\hat{y}}$. *In all examples, the mesh is assumed preexistent below the polygonal perimeter.*

Remark 6.5. The local mesh-generation step above provides no provable guarantee for the quality of the resulting triangles (aspect ratio, minimum angle, etc.). Nevertheless, we note that

1. in practice, the generated mesh is fairly well behaved (e.g., in the example considered in section 7, the minimum angle present in the mesh is $> 18°$ and the vast majority of triangles have minimum angles $> 30°$.)
2. the aspect ratio of the constructed triangles does not directly affect the quality of manifold-approximation. Every $Considered$ mesh-point $\boldsymbol{y}$ is computed using all the segments in $NF(\boldsymbol{y})$, not only the immediately adjacent triangles.
3. additional postprocessing procedures (*diagonal swapping* and *mesh smoothing*) can be used to improve the aspect ratios once the manifold is constructed. See [27] for further details.
4. our mesh-generation method requires $O(1)$ operations for each *Accepted* point. More sophisticated (and more computationally expensive) mesh-generation procedures can be used to obtain triangles with guaranteed minimum angles. For example, mesh-quality guarantees can be obtained for Delaunay triangulation methods [5] and for hybrid advancing-front/Delaunay triangulation methods [29].
5. our method clearly uses the fact that $k = 2$. For the general case $k > 2$, the mesh extension step would require building a (local) simplicial complex in the manifold tangent space compatible with the current polytope boundary (i.e., $AcceptedFront$). This construction has to be performed only locally and no mesh-quality (aspect ratio) guarantees are required. Thus, any hypersurface meshing method can be used (e.g., [4, 17]).

6.6. Stopping criteria. The stopping criterion used in our implementation is the unavailability of any $Considered$ points with $\sigma(\boldsymbol{y}) \leq \Sigma$, where Σ is a prespecified (max-distance-along-the-trajectory) parameter. Using the heap-sort data structure, the criterion can be checked by a single comparison: the algorithm stops as soon as $SortValue(\bar{\boldsymbol{y}}) > \Sigma$, where $\bar{\boldsymbol{y}}$ is the current first element on the heap.

We note that other stopping criteria (Euclidean or geodesic distance, time-along-trajectory, total number of simplexes, etc.) can be used independently of the chosen $SortValue$.

6.7. Algorithm features and possible optimizations. The algorithm we have described has the worst-case computational complexity of $O(R_{NF} N \log K)$, where N is the total number of mesh-points, R_{NF} provides the maximum number of recomputations for a $Considered$ point, and K is the maximum number of mesh-points marked $Considered$ at the same time (typically, $\approx \sqrt{N}$). This is different from the $O(N)$ complexity of the explicit methods of section 2. Nevertheless, this additional cost is justified since it results in a reduction of discretization errors; see, for example, Figure 6 and the discussion of local truncation errors in section 4.1. In addition, the overall computational efficiency of our method is much better since the "constant coefficients terms" in the computational cost are largely dependent on the geometry of the manifold rather than on the geometric stiffness of the vector field (see Remark 2.1).

Remark 6.6. For $k > 2$, a generalization of the algorithm described above will have the same asymptotic complexity of $O(R_{NF} N \log K)$. However, a (constant factor) increase in cost appears for $k < n-1$. In that case, a system of $(n-k)$ PDEs has to be solved to update each $Considered$ point (section 3). The geometric argument in section 4 shows that solving the discretization of that system requires an $(n-k)$-dimensional Newton–Raphson method. This contrasts our method with the approaches introduced in [18, 21, 7], for which the computational cost of updating a single marker increases with the manifold dimension.

Remark 6.7. Given our method of construction, the manifold-approximation always contains the (approximate) trajectory of each already *Accepted* mesh-point. (This stems from the upwinding criteria and is independent of our choices of $SortValue$ and/or stopping criteria.) That property is similarly possessed by the methods introduced in [18] and [6], but not by those in [15] and [20].

Remark 6.8. Our algorithm may terminate before Σ is actually reached: given the particular choices for the method-parameters (desired mesh-scale Δ, initialization radius R_{init}, and $NearFront$ radius R_{NF}), it might be impossible to obtain an accurate (upwinding-condition-satisfying) update for any of the current $Considered$ points. Such $Considered$ points will have $SortValue = +\infty$ and, for the sake of efficiency, will not be placed onto the heap-sort. We note that such "early termination" can be easily determined in $O(K)$ operations by checking if $\sigma(\boldsymbol{y^j}) + \Delta << \Sigma$ for any $\boldsymbol{y^j}$ on the final $AcceptedFront$. For the examples in this paper, the suitable parameter values were obtained empirically. A better implementation would address this adaptively: R_{NF} can be increased and/or Δ can be decreased (by refining the current $AcceptedFront$) whenever a possibility of early termination is detected. In addition, a valuable extension would be to vary these parameters automatically based on the detected information about the manifold geometry (e.g., curvature), the vector field's tangency to $AcceptedFront$, and on the current error estimate. We note that such adaptive versions are already available for some of the prior methods mentioned in section 2 (see, e.g., [21]).

7. Example: The Lorenz system.

We consider the classical example of the Lorenz system [23]:

$$\begin{aligned} x' &= \varsigma(y-x); \\ y' &= \rho x - y - xz; \\ z' &= -\beta z + xy; \end{aligned} \tag{14}$$

with the canonical parameter values $\varsigma = 10$, $\beta = \frac{8}{3}$, and $\rho = 28$.

In this case, the system has three fixed points: the origin and $(\pm 6\sqrt{2}, \pm 6\sqrt{2}, 27)$. The eigenvalues for the Jacobian at the origin are $\lambda \approx -22.8$, -2.67, 11.8; thus, the origin has a two-dimensional stable manifold. The ratio of the eigenvalues suggests (at least locally) the geometric stiffness similar to that encountered in (3).

In addition, the stable manifold of the origin has complicated geometry: it spirals into the famous "butterfly-like" chaotic attractor and also twists around the z-axis; see Figure 9. As a result, it became a de facto standard for testing methods for the invariant manifold-approximation (e.g., compare with [22, 15, 7, 16]).

We initialize the $AcceptedFront$ by subdividing the circle of radius $R_{init} = 2$ around the origin in $E^s(0)$ into $N_0 = 21$ segments (i.e., $\Delta = 0.6$). We start by placing 21 "hanging-simplexes" into the list of $Considered$s and proceed as described in section 5. The calculation

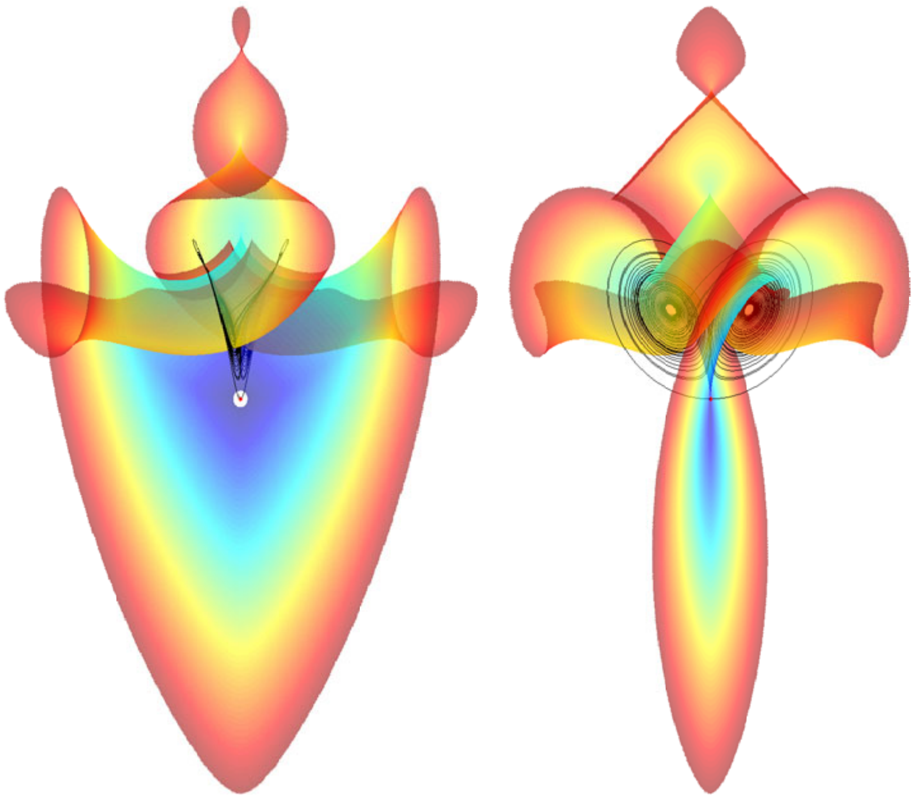


Figure 9. *The invariant manifolds of the origin (two views; rotated around z-axis). The stable manifold (displayed semitransparent) is computed up to $\Sigma = 120$ and the color indicates the trajectory-arc-length σ. The unstable manifold (in black) is computed by integrating initial conditions in $E^u(0)$ forward in time.*

stops once we *Accept* everything with the trajectory-arc-length less than the specified Σ. For the computation in Figure 10, R_{NF} was set to 4Δ; the resulting mesh contained 116,082 mesh-points and 230,011 simplexes.

Based on the visual and numerical evidence, the produced triangulated surfaces seem to converge to $\mathrm{W}^s_\Sigma(0)$ as the accuracy parameters (R_{init} and Δ) tend to zero. Aside from comparing Figures 9 and 10 with those in [22], etc., we also note the indirect evidence of two sample trajectories appearing to lie on the manifold in Figure 10. These trajectories are obtained by integrating backward in time the initial conditions $\pm\varepsilon\boldsymbol{e}$, where ε is small relative to R_{init} and $\boldsymbol{e}$ is a unit eigenvector corresponding to the eigenvalue $\lambda \approx -22.8$. Animated movies visualizing the growth and structure of this manifold are available at

`http://www.math.cornell.edu/~vlad/manifold_movies/lorenz.html`.

Based on a $O(\Delta^2)$ local truncation error of the discretized equation (7), we expect the first-order convergence of the approximation to $W^s_\Sigma(0)$. Unfortunately, there is no known closed-form parameterization for this manifold, which makes computing global approximation errors difficult. Even computing the distance between two such triangulated surfaces (obtained for different Δ's) is not a trivial task. We rely on the examples of section 9 to numerically test the order of convergence of our method.

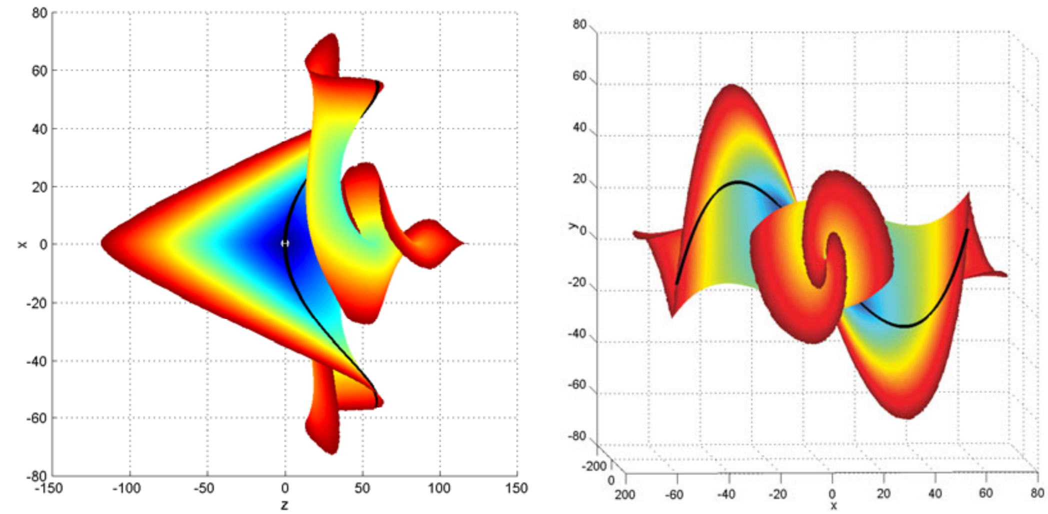


Figure 10. *Stable manifold of the origin computed up to the trajectory-arc-length $\Sigma = 120$. The color indicates σ. Two sample trajectories are shown for verification purposes.*

8. Example: Pendula coupled by torsion.

To demonstrate the applicability of our method to constructing invariant manifolds of higher codimension, we consider here a test problem of two simple pendula coupled by a torsional spring:

$$\begin{aligned} \psi_1''(t) &= -\sin(\psi_1(t)) + \varepsilon(\psi_2(t) - \psi_1(t)), \\ \psi_2''(t) &= -\sin(\psi_2(t)) + \varepsilon(\psi_1(t) - \psi_2(t)). \end{aligned} \tag{15}$$

This problem is discussed in detail in [3]; here, we reproduce only some basic properties of the system.

In (15), ψ_i is the angular position of the ith pendulum, the full state of the system can be recorded as $(\psi_1, \psi_2, \psi_1', \psi_2')$, and the full phase-space is therefore four-dimensional. As written above, the system is conservative, with the total energy given by

$$E = \frac{(\psi_1')^2}{2} + \frac{(\psi_2')^2}{2} - \cos(\psi_1) - \cos(\psi_2) + \frac{\varepsilon\,(\psi_1 - \psi_2)^2}{2}. \tag{16}$$

The constant ε corresponds to a scaled Hooke's law coefficient and we are interested in investigating the system for $\varepsilon \ll 1$ (e.g., $\varepsilon = 0.01, 0.05$). We would like to construct the invariant manifolds of the saddle point at $\boldsymbol{z^0} = (\pi, \pi, 0, 0)$ (i.e., both pendula standing upright with zero angular velocity). The eigenvalues of the Jacobian matrix are $\pm\sqrt{1-2\varepsilon}$ and ± 1; thus, both stable and unstable manifolds are two-dimensional and there are no multiple time-scales in the linearized system near $\boldsymbol{z^0}$. However, the energy level $E = 2$ corresponding to this saddle is singular: it contains both stable and unstable manifolds of all the equilibria of the form $\boldsymbol{z^m} = ((2m+1)\pi, (2m+1)\pi, 0, 0)$. At the same time, for $i \neq j$, this energy level *does not* contain any points of the form $((2i+1)\pi, (2j+1)\pi, \cdot, \cdot)$ —because of the torsional spring, the potential energy at those points is higher than $E(\boldsymbol{z^0})$.

Figure 11 shows the projections of the entire energy level and of sample trajectories in $W^u(\boldsymbol{z^0})$ into the configuration plane (ψ_1, ψ_2) for different values of ε. The configuration space has a periodic structure (the behavior at (ψ_1, ψ_2) is the same as at $(\psi_1+2\pi m, \psi_2+2\pi m)$). The uncoupled system (for $\varepsilon = 0$) is doubly periodic and, as a result, for small ε the configuration plane clearly has a cellular structure. The *cells* are the squares whose vertices are at the points $((2m+1)\pi, (2n+1)\pi)$ and whose boundaries correspond to the state where one of the pendula is upright and the dynamics is especially sensitive.

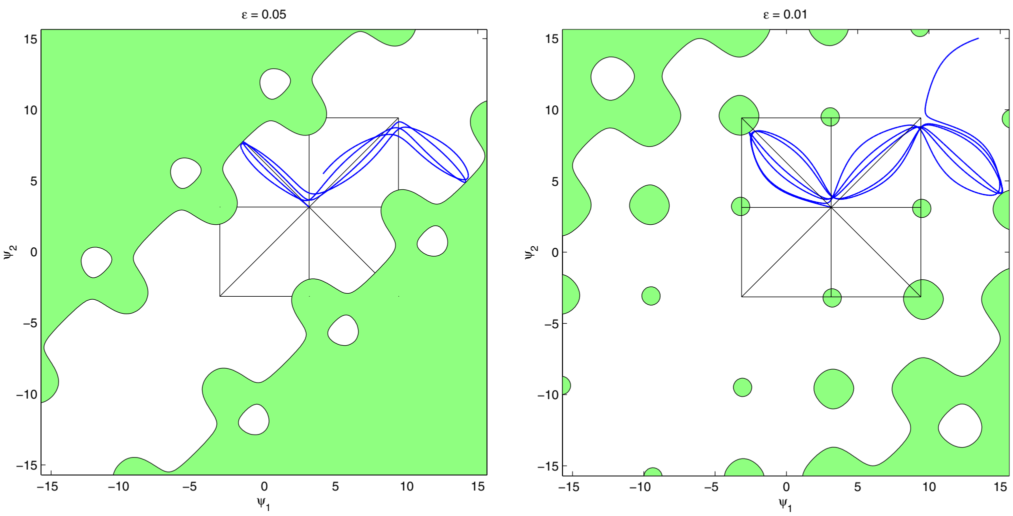


Figure 11. *Projection into the configuration space: the energy level and a single (typical) orbit in* $W^u(\boldsymbol{z^0})$ *for* $\varepsilon = 0.05$ *and* $\varepsilon = 0.01$*. The light green regions are unattainable due to the energy conservation.*

Several types of connecting orbits inside the energy level can be determined and are very useful in assessing the accuracy of the invariant manifold computations. For example, there are heteroclinic trajectories connecting each $\boldsymbol{z^m}$ with $\boldsymbol{z^{m\pm 1}}$ (corresponding to the pendula moving in unison) and homoclinic orbits lying along the "antidiagonals" $\psi_1 + \psi_2 = (4m + 2)\pi$ (corresponding to the pendula departing from z^m in opposite directions and moving in symmetry until the spring pulls them back).

Here, we present several views of $W^u(\boldsymbol{z^0})$ for $\varepsilon = 0.01$. Animated movies visualizing the growth and structure of the manifold for $\varepsilon = 0.1$ are available at

`http://www.math.cornell.edu/~vlad/manifold_movies/pendula.html`.

In our computation, we initialized the *AcceptedFront* on a circle of radius $R_{init} = 0.5$ in $E^u(\boldsymbol{z^0})$, used $\Delta = 0.1$, and computed the manifold up to $\sigma_{\max} = 15.5$. We note several algorithmic differences from the previous example. An energy conserving method could be built to essentially reduce the search-space to a single dimension (i.e., except at $\boldsymbol{z^m}$, the energy level is three-dimensional and the invariant manifolds have codimension 1 inside it). Instead, we have chosen to implement the method in the full four-dimensional phase space, thus illustrating the construction of a manifold of codimension 2. A system of two quasi-linear PDEs is solved to obtain each *Considered* point $\boldsymbol{y}$ (see (5)); the solution to the discretized nonlinear system is obtained by a standard two-dimensional Newton–Raphson method (e.g., see [28]). The resulting triangulated mesh approximates $W^u(\boldsymbol{z^0})$. However, once the *AcceptedFront* is sufficiently close to $\boldsymbol{z^{\pm 1}}$, the numerical losses in energy result in "retracting" along the

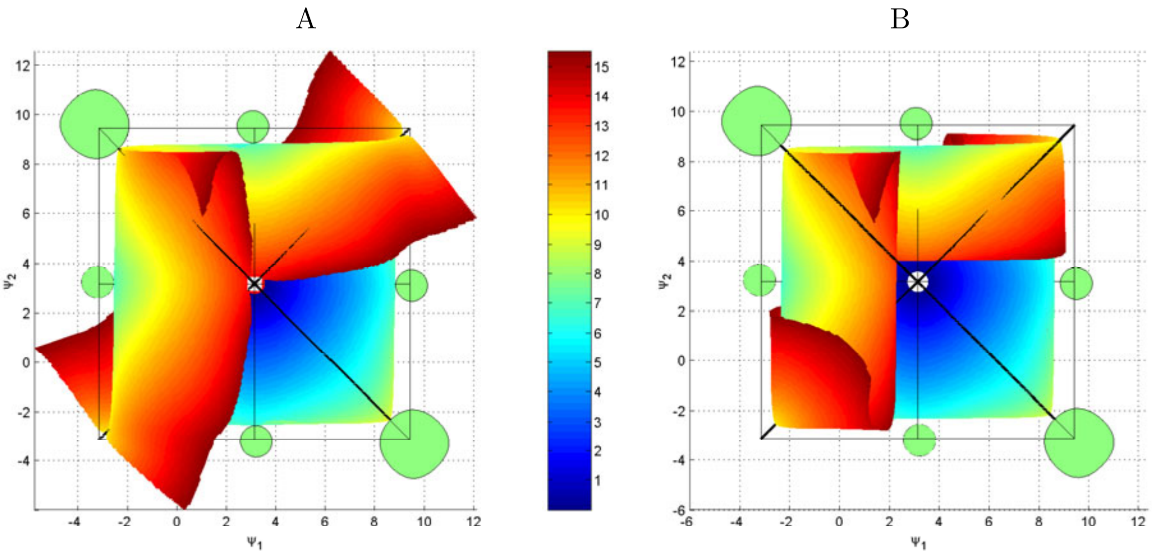


Figure 12. *Projection of $W^u(\boldsymbol{z^0})$ onto the (ψ_1, ψ_2) plane ($\Sigma = 15.5$; coloring indicates σ; a total of $77{,}500$ mesh-points) computed* with *(left) and* without *(right) "projection onto the energy level" procedure. Black lines indicate the "cell" and homoclinic/heteroclinic trajectories. The light green "ink spots" indicate regions unattainable due to the energy conservation (as in Figure 11). The right picture clearly shows the loss of energy in the process of computation.*

unstable manifolds $W^u(\boldsymbol{z^1})$ and $W^u(\boldsymbol{z^{-1}})$; see Figure 12B. This "folding onto itself" is a numerical artifact rather than a feature of $W^u(\boldsymbol{z^0})$.

To handle this problem, we implement an additional step of projection onto the energy level: Immediately before a *Considered* mesh-point $\boldsymbol{y}$ is *Accepted*, we solve an initial value problem

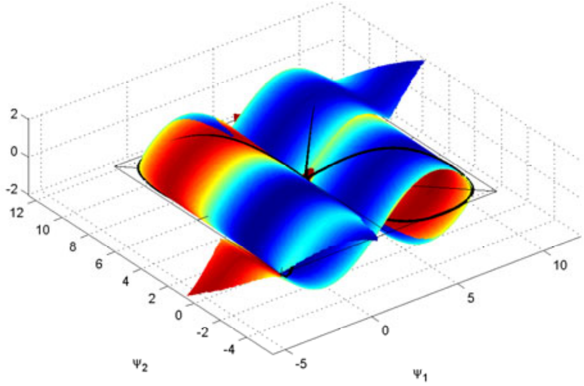
$$\boldsymbol{y}_p'(t) = \operatorname{sign}(2 - E(\boldsymbol{y}))\, \nabla E(\boldsymbol{y}_p(t)), \qquad \boldsymbol{y}_p(0) = \boldsymbol{y},$$

until the first intersection $\boldsymbol{y}_p(\tau)$ with the level set $E = 2$. If that point is within $(\Delta/10)$ from $\boldsymbol{y}$, we set $\boldsymbol{y} = \boldsymbol{y}_p(\tau)$ and continue as described in section 6; otherwise, the algorithm terminates since the local energy loss after solving the PDE is considered too large.

This results in a much better approximation of $W^u(\boldsymbol{z^0})$; see Figure 12A. Unfortunately, this projection procedure becomes unstable near all $\boldsymbol{z^m}$'s since both $W^u(\boldsymbol{z^m})$ and $W^s(\boldsymbol{z^m})$ lie in the same energy level $E = 2$, which becomes singular at each $\boldsymbol{z^m}$. Thus, our implementation artificially stops the manifold from growing too close to those points; i.e., tentative triangles are not added to segments of $AcceptedFront$ which are within $R_{restrict} = 0.3$ from $\boldsymbol{z^m}$.

Figure 13 shows two homoclinic orbits of $\boldsymbol{z^0}$ and two heteroclinic orbits connecting $\boldsymbol{z^0}$ to $\boldsymbol{z^1}$ and $\boldsymbol{z^{-1}}$. These trajectories appear to lie on the computed manifold-approximation, indirectly confirming convergence to $\mathrm{W}^u_\Sigma(\boldsymbol{z^0})$. However, a direct verification of convergence for this example is hard due to the lack of analytic formulae for $W^u(\boldsymbol{z^0})$.

9. Example: An egg carton surface. Finally, in order to test the rate of convergence numerically, we consider a simple example for which the invariant manifold is a priori known.



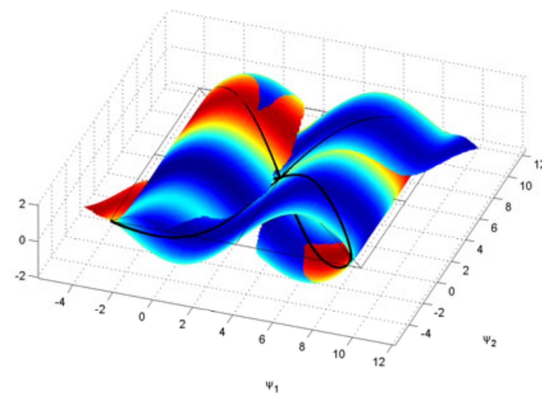


Figure 13. *Two different (rotated) views of the projection of $W^u(\boldsymbol{z^0})$ into the $(\psi_1, \psi_2, \psi_1')$ space. Conservation of energy is enforced by the projection procedure. Coloring is used to indicate the fourth coordinate (ψ_2') ranging from -2.07 (blue) to 2.07 (red). The seeming self-intersection is a side effect of the three-dimensional projection side effect. The thin black lines indicate the "cell." The thick black lines indicate sample homoclinic and heteroclinic trajectories.*

Given a smooth function $g(x, y)$, we consider a system

$$\begin{aligned} x' &= \eta_1 x; \\ y' &= \eta_2 y; \\ z' &= -\mu z + \mu g(x, y) + \eta_1 x g_x(x, y) + \eta_2 y g_y(x, y). \end{aligned} \tag{17}$$

If η_1, η_2, and μ are positive, then the point $(0, 0, g(0, 0))$ is a saddle and the graph of $g(x, y)$ is its unstable manifold. For testing purposes, we have chosen an "egg carton" function $g(x, y) = 0.27 \sin(2\pi x) \sin(2\pi y)$; see Figure 14. Of course, the choice of (η_1, η_2, μ) also influences the computational error; e.g., a bigger μ will obviously make this an easier problem since μ is the rate at which all trajectories are pushed toward the manifold.

For every mesh-point (x, y, z), the approximation error $\mathcal{E}$ is the distance to the manifold surface. An upper bound is readily available as $\mathcal{E}(x, y, z) \leq |z - g(x, y)|$ and can be used to compute the bound on L_2 and L_∞ errors for the entire mesh. In these tests we used $\Sigma = 1.5$ and two different values for μ (1 and 1/4). All computations were repeated for the "isotropic" case $\eta_1 = \eta_2 = 1$ (see Figure 15) and for the "anisotropic" case $\eta_1 = \eta_2/5 = 1$ (see Figure 16). As expected, we observe a quadratic growth of the number of mesh-points N and a linear decay of the approximation error in all of the examples.

10. Conclusions. We have introduced a fast algorithm for approximating invariant manifolds of saddle points of the vector fields in R^n. The chief advantage of this method is its efficiency: all the examples presented in sections 7 and 8 take under 90 seconds to compute on a Pentium III 850 MHz processor with 256Mb RAM. Our approach is new and many related issues remain open. Possible directions for future work include higher-order methods, error bounds and estimates (possibly using an interval arithmetic implementation), adaptive and parallel methods, exploration of robustness under parameter variation, and proofs of convergence. The previously available methods described in section 2 are more developed

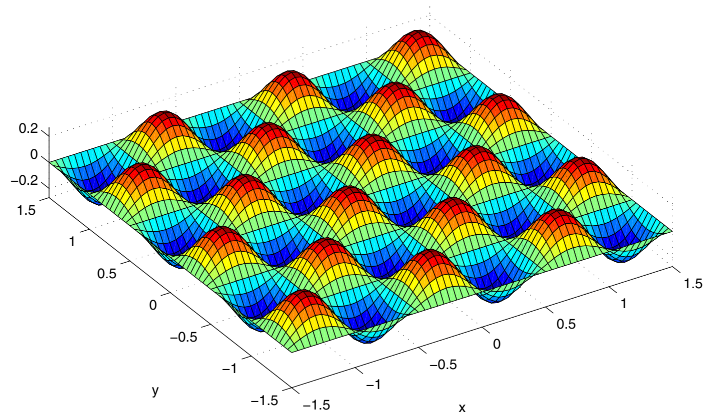
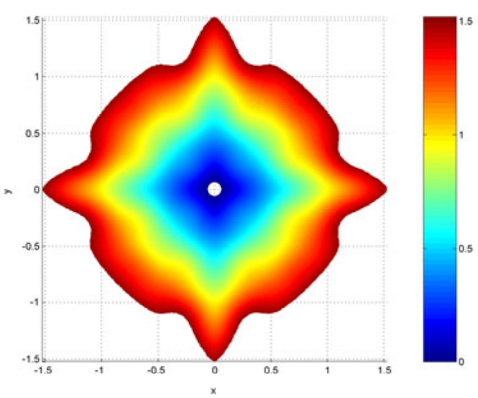


Figure 14. *An "egg carton" function $g(x, y) = 0.27 \sin(2\pi x) \sin(2\pi y)$.*



Parameters		$\mu = 1$		
Δ	R_{init}	N	L_2 Error	L_∞ Error
$\Delta_0 = 0.05$	$r_0 = 0.2$	3940	0.034316	0.103143
$\Delta_0 \times 2^{-1}$	$r_0 \times 2^{-1}$	14039	0.016671	0.056432
$\Delta_0 \times 2^{-2}$	$r_0 \times 2^{-2}$	53271	0.008582	0.028589
$\Delta_0 \times 2^{-3}$	$r_0 \times 2^{-3}$	208626	0.004168	0.015052
$\Delta_0 \times 2^{-4}$	$r_0 \times 2^{-4}$	829158	0.002086	0.007570

Parameters		$\mu = 0.25$		
Δ	R_{init}	N	L_2 Error	L_∞ Error
$\Delta_0 = 0.05$	$r_0 = 0.2$	3894	0.046520	0.152055
$\Delta_0 \times 2^{-1}$	$r_0 \times 2^{-1}$	13936	0.023853	0.081145
$\Delta_0 \times 2^{-2}$	$r_0 \times 2^{-2}$	53159	0.012685	0.043746
$\Delta_0 \times 2^{-3}$	$r_0 \times 2^{-3}$	208502	0.005986	0.021406
$\Delta_0 \times 2^{-4}$	$r_0 \times 2^{-4}$	828884	0.002969	0.011011

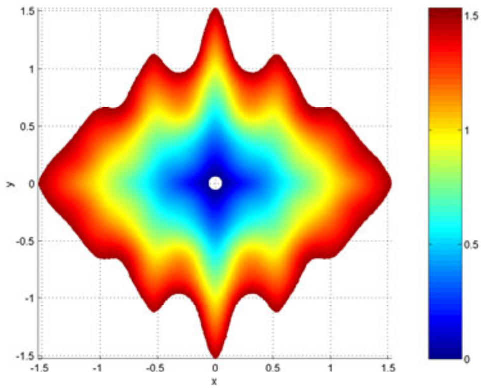
Figure 15. *Isotropic case ($\eta_1 = 1$, $\eta_2 = 1$). The (x-y)-projection of the mesh (color indicates trajectory-arc-length σ) and the table of error bounds.*

(with many extensions available), but, to the best of our knowledge, are substantially more time-consuming on the problems with multiple time-scales.

The perspective of building the manifold as a collection of simplexes, each of them satisfying a locally posed PDE, is quite general. For example, customized stopping criteria can be used to treat manifolds converging to attracting limit sets. We are also planning to investigate the applicability of our approach to approximating invariant manifolds of saddle-type cycles (see [19] and [26] for the existing methods).

Our current implementation relies on $k = 2$ for the local mesh-generation procedure only. We expect that a combination of our approach with robust techniques for higher-dimensional mesh extension will yield fast methods for the general case (see Remark 6.5).

Finally, we note that some of the ideas illustrated above may be useful in the context



Parameters		$\mu = 1$		
Δ	R_{init}	N	L_2 Error	L_∞ Error
$\Delta_0 = 0.05$	$r_0 = 0.2$	3629	0.038158	0.090629
$\Delta_0 \times 2^{-1}$	$r_0 \times 2^{-1}$	13429	0.018711	0.049707
$\Delta_0 \times 2^{-2}$	$r_0 \times 2^{-2}$	51466	0.009302	0.024855
$\Delta_0 \times 2^{-3}$	$r_0 \times 2^{-3}$	204446	0.004601	0.011553
$\Delta_0 \times 2^{-4}$	$r_0 \times 2^{-4}$	815410	0.002374	0.006496

Parameters		$\mu = 0.25$		
Δ	R_{init}	N	L_2 Error	L_∞ Error
$\Delta_0 = 0.05$	$r_0 = 0.2$	3621	0.040681	0.127953
$\Delta_0 \times 2^{-1}$	$r_0 \times 2^{-1}$	13425	0.020136	0.059774
$\Delta_0 \times 2^{-2}$	$r_0 \times 2^{-2}$	52066	0.010428	0.026192
$\Delta_0 \times 2^{-3}$	$r_0 \times 2^{-3}$	205321	0.005296	0.013102
$\Delta_0 \times 2^{-4}$	$r_0 \times 2^{-4}$	815905	0.002631	0.007336

Figure 16. *Anisotropic case ($\eta_1 = 1$, $\eta_2 = 5$). The (x-y)-projection of the mesh (color indicates trajectory-arc-length σ) and the table of error bounds.*

of prior methods, which grow the manifold as a collection of $(k-1)$-dimensional topological spheres. In particular, we believe that the method defined in [20, 21] can be substantially accelerated by using parts of M_{i+1} as they become available (as opposed to using M_i only and producing the entire M_{i+1} at once). Further speedup can be attained by ordering the computation of markers (first compute those whose trajectories are "the least tangential" to M_i) and using a discretized system-solver instead of the time-consuming shooting methods.

Acknowledgments. The authors would like to thank S. Vavasis, K. Lin, O. Junge, H. Osinga, B. Krauskopf, and R. Sacker.

REFERENCES

[1] R. Altendorfer, R. Ghigliazza, P. Holmes, and D. E. Koditschek, *Exploiting passive stability for hierarchical control*, in Proceedings of the Fifth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2002), P. Bidaud and F. Ben Amar, eds., Professional Engineering Publishing, London, UK, 2002. Available online at http://ai.eecs.umich.edu/CNM/Publications/Published_Articles/Clawar2002.pdf

[2] H. D. Chiang, F. F. Wu, and P. P. Varaiya, *A BCU method for direct analysis of power system transient stability*, IEEE Trans. Power Systems, 9, (1994), pp. 1194–1208.

[3] D. G. Aronson, E. J. Doedel, J. Guckenheimer, and B. Sandstede, *On the Dynamics of Torsion-Coupled Pendula*, in preparation.

[4] M. L. Brodzik, *The computation of simplicial approximations of implicitly defined p-dimensional manifolds*, Comput. Math. Appl., 36, (1998), pp. 93–113.

[5] Paul L. Chew, *Guaranteed-Quality Triangular Meshes*, Tech. report 89-983, Department of Computer Science, Cornell University, Ithaca, NY, 1989.

[6] 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., Progr. Nonlinear Differential Equations Appl. 19, Birkhäuser, 1996, pp. 449–459.

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

[8] L. Dieci and J. Lorenz, *Computation of invariant Tori by the method of characteristics*, SIAM J. Numer. Anal., 32 (1995), pp. 1436–1474.

[9] L. Dieci, J. Lorenz, and R. D. Russell, *Numerical calculation of invariant tori*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 607–647.
[10] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Y. A. Kuznetsov,, B. Sandstede, and X. J. Wang, *AUTO97 : Continuation and Bifurcation Software for Ordinary Differential Equations*, 1997. Available online at ftp://ftp.cs.concordia.ca/pub/doedel/auto/auto.ps.gz
[11] E. J. Doedel, *Private communication*, IMA, October 1997.
[12] K. D. Edoh, R. D. Russell, and W. Sun, *Orthogonal Collocation for Hyperbolic PDEs and Computation of Invariant Tori*, Mathematics Research Report MRR 060-95, Australian National University, Canberra, Australia, 1995.
[13] N. Fenichel, *Persistence and smoothness of invariant manifolds for flows*, Indiana Univ. Math. J., 21, (1971), pp. 193–226.
[14] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields*, Springer-Verlag, Berlin, 1983.
[15] J. Guckenheimer and P. Worfolk, *Dynamical systems: Some computational problems*, in Bifurcations and Periodic Orbits of Vector Field, D. Schlomiuk, ed., NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. 408, Kluwer Academic Publishers, 1993.
[16] M. E. Henderson, *Computing invariant manifolds by integrating fat trajectories*, SIAM J. Applied Dynamical Systems, submitted.
[17] M. E. Henderson, *Multiple parameter continuation: Computing implicitly defined k-manifolds*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 12 (2002), pp. 451–476.
[18] M. E. Johnson, M. S. Jolly, and I. G. Kevrekidis, *Two-dimensional invariant manifolds and global bifurcations: Some approximation and visualization studies*, Numer. Algorithms, 14 (1997), pp. 125–140.
[19] M. E. Johnson, M. S. Jolly, and I. G. Kevrekidis, *The Oseberg transition: Visualization of global bifurcations for the Kuramoto-Sivashinsky equation*, Internat. J. Bifur. Chaos, 11 (2001), pp. 1–18.
[20] B. Krauskopf and H. Osinga, *Two-dimensional global manifolds of vector fields*, Chaos, 9 (1999), pp. 768–774.
[21] B. Krauskopf and H. Osinga, *Global Manifolds of Vector Fields: The General Case*, Applied Nonlinear Mathematics Research Report 99.2, University of Bristol, Bristol, UK, 1999.
[22] B. Krauskopf and H. Osinga, *Visualizing the structure of chaos in the Lorenz system*, Comput. Graphics, 26 (2002), pp. 815–823.
[23] E. N. Lorenz, *Deterministic nonperiodic flows*, J. Atmospheric Sci., 20 (1963), pp. 130–141.
[24] H. Mingyou, T. Küpper, and N. Masbaum, *Computation of invariant tori by the Fourier methods*, SIAM J. Sci. Comput., 18 (1997), pp. 918–942.
[25] J. Moser, *On invariant manifolds of vector fields and symmetric partial differential equations*, Differential Anal., Bombay Colloq., (1964), pp. 227–236.
[26] H. Osinga, *Nonorientable manifolds of three-dimensional vector fields*, Internat. J. Bifur. Chaos, 13 (2003), pp. 553–570.
[27] J. Peraire, J. Peiro, and K. Morgan, *Advancing front grid generation*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, Boca Raton, FL, 1999, Chapter 17.
[28] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C,* Cambridge University Press, Cambridge, UK, 1988.
[29] S. Rebay, *Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm*, J. Comput. Phys., 106 (1993), pp. 125–138.
[30] R. J. Sacker, *A new approach to the perturbation theory of invariant surfaces*, Comm. Pure Appl. Math., 18 (1965), pp. 717–732.
[31] S. Fomel and J. A. Sethian, *Fast-phase space computation of multiple arrivals*, Proc. Nat. Acad. Sci., 99 (2002), pp. 7329–7334.
[32] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton-Jacobi equations*, Proc. Nat. Acad. Sci., 98 (2001), pp. 11069–11074.

[33] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations: Theory and applications*, SIAM J. Numer. Anal., 41 (2003) pp. 325–363.

[34] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for hybrid control*, in Proceedings of the 5th International Workshop (HSCC 2002, Stanford, CA, March 25-27, 2002), Lecture Notes in Comput. Sci. 2289, Springer-Verlag, New York, NY, pp. 393–406.

[35] J. Smoller, *Shock Waves and Reaction-Diffusion Equations*, 2nd ed., Springer-Verlag, New York, 1994.

[36] A. Vladimirsky, *Space-Marching Methods for the First-Order Non-Linear PDEs*, in preparation.