International Journal of Bifurcation and Chaos, Vol. 21, No. 7 (2011) 2017–2042
© World Scientific Publishing Company
DOI: 10.1142/S0218127411029604

HIGH-ORDER BISECTION METHOD FOR COMPUTING INVARIANT MANIFOLDS OF TWO-DIMENSIONAL MAPS

ROY H. GOODMAN* and JACEK K. WRÓBEL†
Department of Mathematical Sciences,
New Jersey Institute of Technology,
Newark, NJ 07102, USA
**roy.goodman@njit.edu*
†jacek.wrobel@njit.edu

Received July 14, 2010

We describe an efficient and accurate numerical method for computing smooth approximations to invariant manifolds of planar maps, based on geometric modeling ideas from Computer Aided Geometric Design (CAGD). The unstable manifold of a hyperbolic fixed point is modeled by a piecewise Bézier interpolant (a Catmull–Rom spline) and properties of such curves are used to define a rule for adaptively adding points to ensure that the approximation resolves the manifold to within a specified tolerance. Numerical tests on a variety of example mappings demonstrate that the new method produces a manifold of a given accuracy with far fewer calls to the map, compared with previous methods. A brief introduction to the relevant ideas from CAGD is provided.

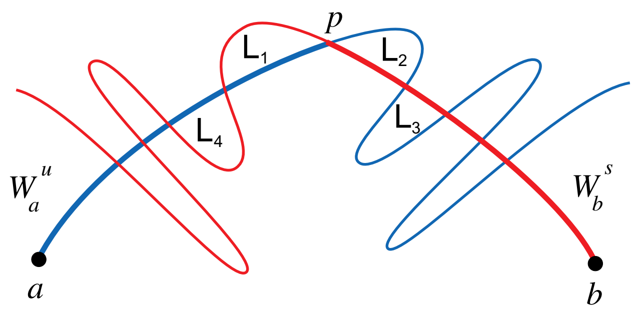
Keywords: Invariant manifold; computer-aided geometric design; numerical algorithm.

1. Introduction

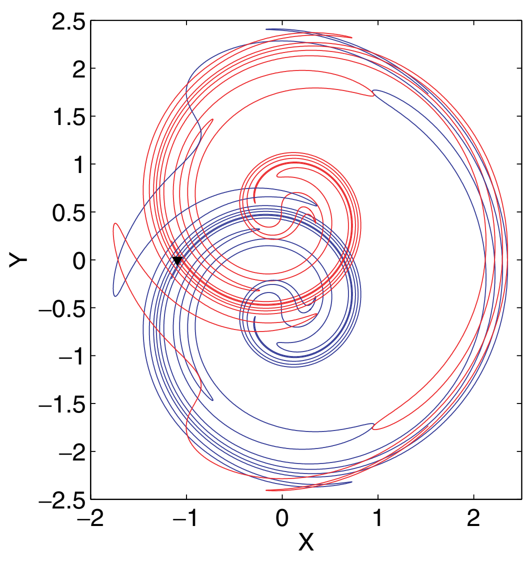
Iterated maps are ubiquitous in the study of dynamics, arising either as models of physical, biological, or economic systems themselves or as reductions of continuous-time dynamics, e.g. as Poincaré maps. Of fundamental importance in understanding these dynamical systems are invariant manifolds (stable or unstable) emanating from fixed points and periodic orbits. These manifolds act as barriers between different regions of the phase space and exert a significant influence on the dynamics through their topology. Except in some rare cases, such a manifold cannot be expressed as a closed form parametric curve, nor as the level set of some function, and therefore must be approximated numerically. It typically forms a so-called "tangle" in which the manifold doubles back on itself repeatedly, with segments of very high curvature connected to other segments where the curvature is more modest.

The extreme stretching and folding of these curves gives rise to chaotic dynamics. For the well-known Hénon map [Hénon, 1976], a dissipative and chaotic dynamical system, the *strange attractor* is equal to the closure of W^{u}, so constructing it provides a way to approximate the strange attractor and thus to understand the long-time behavior of the system. Similar reasoning applies to other dissipative chaotic systems possessing an attractor.

Two examples of one-dimensional invariant manifolds are shown in Fig. 1. Such a manifold is generally infinite in extent, so any computation will approximate only a finite portion. In Fig. 1(a), we consider two regions defined as the areas above and below the curve *apb*. In a model of fluid mixing the four "lobes" $L_1, \ldots, L_4$, are known as turnstiles, and can be used to quantify how fluid moves between the upper and lower regions, according to the theory



(a)



(b)

Fig. 1. (a) Schematic of stable/unstable manifolds (red/blue). (b) Stable/unstable manifolds (red/blue) of a fixed point computed using the algorithm of [Carter, 2004].

of *phase-space transport* as in [Rom-Kedar, 1990, 1994]; see also the work [Meiss, 1997]. More recent work by Collins, and by Mitchell and Delos has shown how to use the information contained in the intersections of the stable and unstable manifolds to construct symbolic dynamics and obtain a fuller topological understanding of these systems [Collins, 2002; Mitchell, 2009; Mitchell & Delos, 2006]. Figure 1(b) shows the stable and unstable manifolds for the map defined in [Goodman, 2008] and demonstrates the intricacy of such curves in an area-preserving example.

This paper presents a new numerical method for computing the one-dimensional unstable manifolds of hyperbolic fixed points of smooth maps from the plane to itself. The method is based on simple concepts developed in the field of computer-aided geometric design (CAGD). Several other methods exist in the literature, which we review in Sec. 5.1 and in the concluding discussion. These algorithms are adaptive in the sense that more points are placed in regions of high curvature than in those with modest curvature. Our tests found that these methods placed far more points than needed in these high-curvature regions, and this led to slower than optimal convergence, and, in some cases, failure of the method to terminate. We test some of these methods against our new ones, using more stringent tests than we have seen previously applied, and find significant improvement.

The remainder of this paper is organized as follows. Section 2 contains more thorough background information about unstable manifolds. In Sec. 3, we introduce the model problem of drawing a parametric curve along with a few methods for parameterizing unstable manifolds. Section 4 gives an introduction to the tools from CAGD that are at the heart of the algorithms, while a more technical discussion of these ideas is presented in Appendix A. Topics covered include piecewise linear interpolation, Bézier curves, and Catmull–Rom splines. Section 5 contains descriptions of adaptive methods for rendering parametric curves. We first show existing methods due to Hobson and Carter and then describe two new methods. In Sec. 6, we perform numerical tests of the geometric tools introduced earlier, especially Catmull–Rom splines, as well as tests of the proposed methods for rendering parametric curves. Section 7 contains a further description of the implementation of our proposed methods in the context of computing invariant manifolds. In Sec. 8, we perform various numerical tests showing convergence of the methods. Section 9 contains a summarizing discussion of the advantages of the algorithms presented here as well as a partial road map for future work.

Finally, there are two appendices. The first, Appendix A, contains a detailed introduction to the CAGD tools used in this paper. The second, Appendix B, contains links to supplementary material, including working MATLAB codes implementing the algorithms described in the paper.

2. Background

A discrete-time iterated map is a dynamical system $\mathbf{x}_{j+1} = f(\mathbf{x}_j)$ where, for simplicity, we assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is diffeomorphism, and as smooth as we need. For the present work, we set $n = 2$. When considering the system simply as a diffeomorphism and not as a dynamical system, we write $\mathbf{x}' = f(\mathbf{x})$. We assume that f has a hyperbolic fixed point $\mathbf{x}^*$, i.e. $f(\mathbf{x}^*) = \mathbf{x}^*$ such that the linearized matrix $\mathrm{D}f(\mathbf{x}^*)$ has two real eigenvalues λ_s and λ_u satisfying $0 < |\lambda_s| < 1 < |\lambda_u|$. For simplicity assume both are positive. These assumptions assure that the map is orientation-preserving and has one-dimensional stable and unstable manifolds of $\mathbf{x}^*$ in $\mathbb{R}^2$, invariant under f.

The stable manifold,

$$W^{\mathrm{s}}(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^2 : f^k(\mathbf{x}) \to \mathbf{x}^* \text{ as } k \to \infty\},$$

is defined as the set of points which approach $\mathbf{x}^*$ in forward iterates of the map.[1] The manifold is tangent to the stable eigendirection of the linearized system at $\mathbf{x}^*$ and its global extension can be derived by applying the inverse mapping f^{-1} to the local segment. The unstable manifold,

$$W^{\mathrm{u}}(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^2 : f^{-k}(\mathbf{x}) \to \mathbf{x}^* \text{ as } k \to \infty\},$$

is defined as the set of points which approach $\mathbf{x}^*$ in backward iterates of the map. The assumptions on the eigenvalues ensure this convergence is exponential. The manifold is tangent to the unstable eigendirection of the linearized system at $\mathbf{x}^*$ and its global extension can be derived by applying the forward mapping to a local segment.

Most numerical methods constructed to compute invariant manifolds use the same basic idea: the global structure of individual branches of the unstable manifold is found by repeatedly applying the mapping to an existing segment of the manifold. The stable manifold can similarly be found by iterating f^{-1}. Methods also exist for computing W^{s} when f^{-1} is not available in closed form [Kostelich *et al.*, 1996].

Consider two points $\mathbf{x}, \mathbf{y} \in W^{\mathrm{u}}$. Let $W^{\mathrm{u}}[\mathbf{x}, \mathbf{y}]$ denote the closed segment of W^{u} connecting $\mathbf{x}$ to $\mathbf{y}$. For any given point $\mathbf{x}_0 \in W^{\mathrm{u}}$, the set of its images $\{\mathbf{x}_n = f^n(\mathbf{x}_0) : n \in \mathbb{Z}\}$ partition W^{u} into a family of finite curve segments disjoint except for their endpoints. We refer to the closed connected component of W^{u} between $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ as the nth primary segment $\mathbf{U}_n$, i.e.

$$\mathbf{U}_n = W^{\mathrm{u}}[\mathbf{x}_n, f(\mathbf{x}_n)].$$

A single branch $\mathbf{U}$ of the unstable manifold associated with the fixed point $\mathbf{x}^*$ can be constructed as the union of the primary segments $\mathbf{U}_n$. Therefore, we can write

$$\mathbf{U} = \bigcup_{n=-\infty}^{\infty} \mathbf{U}_n,$$

where the branch is determined by the choice of the initial primary segment $\mathbf{U}_0$. This is obtained by local analysis and it is usually taken very close to $\mathbf{x}^*$; see Fig. 2. The short portion of the branch of W^{u} between $\mathbf{x}^*$ and the initial primary segment can be generated by backward iterates. It is however of little interest.

The problem of computing a finite portion of an unstable manifold can be reduced to that of simply computing a parametric curve in the following manner. Given an already computed segment $\mathbf{U}_n$ that has been endowed with parameterization

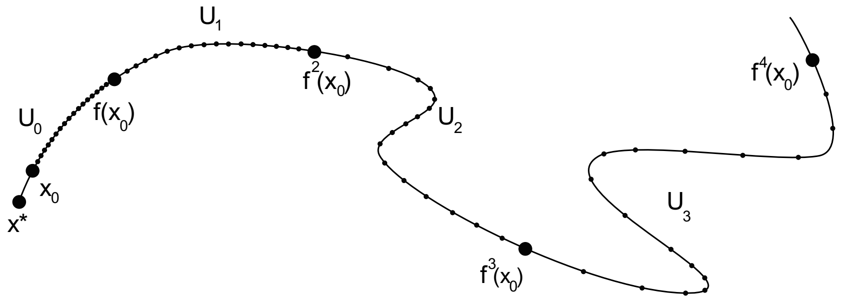


Fig. 2. (Schematic) Union of primary segments forming a portion of the unstable manifold.

[1]A more precise definition of these manifolds requires exponential convergence. This condition is fulfilled for any system satisfying the conditions given on λ_s and λ_u.

$\mathbf{U}_n = \mathbf{U}_n(t)$, $a \leq t \leq b$, then the next segment is simply

$$\mathbf{U}_{n+1}(t) = f(\mathbf{U}_n(t)), \tag{1}$$

so that $\mathbf{U}_{n+1}(t)$ is a parametric curve depending on the same parameter. We further discuss how to choose the parameterization in Sec. 3.

Clearly any numerical method employing this idea by using the same number of points to approximate the initial primary segment $\mathbf{U}_0$ and all its forward images will have several disadvantages. First, the distribution of points along each segment is not controlled. A few iterations of the initial segment may produce very closely spaced points in some regions while large gaps between points occur elsewhere. Next, the length of a primary segment tends to increase rapidly with the number of iterations of the initial segment. The number of points required to resolve a later segment is generally much greater than that required to resolve previous segments; see Fig. 2.

We focus our work on adaptive methods for computing invariant manifolds. These methods are able to adapt the distribution and number of points on each segment. This avoids large gaps between successive points on a segment without placing too many points on that segment. It can also avoid using far too many points in the shorter segments while still resolving the longer segments. Our goal is to generate an approximation of the manifold which is smoothly resolved with a minimum number of points.

3. Model Curves and Parameterization

3.1. *Parametric curves*

Motivated by (1), we delay considering the problem of computing an unstable manifold and instead focus on the simpler problem of drawing a parametric curve

$$\gamma = \{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = g(t), a \leq t \leq b\}. \tag{2}$$

To discuss what "drawing a curve" means, it is useful to first discuss the idea of "geometric modeling". Any computational algorithm must replace the continuous mathematical object γ by a finite collection of data that can be stored and manipulated by a computer. These tools were developed as part of the field of CAGD and are at the heart of modern computer graphics, animation and CAD programs.

Most existing methods for calculating invariant manifolds do not explicitly discuss the idea of geometric modeling, but all implicitly choose a model in order to construct an algorithm. In the cited references, the curve is modeled by a "discrete curve", a sequence of points connected by a linear interpolant. The question addressed by the existing algorithms, then, is how best to generate a finite set of parameter values such that the model curve satisfies some analytic or aesthetic criteria. We discuss previous algorithms in Sec. 5.1. Similar existing methods for generating two-dimensional invariant manifolds are likewise based on linear (planar) interpolation of points on a surface. As loose analogy, algorithms based on linear interpolation have local approximation error of $O(\Delta^2)$ where Δ is the distance between two consecutive interpolation points — the equivalent of the forward Euler schemes students first learn for the solution of ODE initial value problems. The proposed research would, stretching this metaphor, bring this to the level of second or third-order Runge–Kutta, with local errors scaling as Δ^3 or Δ^4 — this is made more precise in the next section.

In addition to drawing a curve that "looks nice" at screen resolution or upon "zooming in" to examine the features of a curve, there are analytic and topological criteria that our approximate or model curve should meet. For example, the unstable manifold W^{u} cannot have any self-intersections. The algorithms below would need to be modified to meet these additional criteria.

3.2. *Parameterization*

The spline interpolation problem in numerical analysis is usually stated with given points $\{\mathbf{x}_0, \ldots, \mathbf{x}_n\}$ and corresponding parameter values $\{t_0, \ldots, t_n\}$. In applications such as computer graphics, the parameter values are rarely provided and therefore must be chosen somehow. Chapter 9 of the textbook [Farin, 2002], dedicates a fair amount of discussion on how best to pick the parameter values. Two commonly used parameterizations defined inductively by $t_k = k$ and *the accumulated chord length parameterization*: one inductively defines $t_0 = 0$ and $t_k - t_{k-1} = \|\mathbf{x}_k - \mathbf{x}_{k-1}\|$, which gives a crude approximation to arc length parameterization. The first method often works poorly because it ignores the geometry of the points. The second method spaces the knots proportionally to the distance between points and usually produces better results, although not in all cases.

Another parameterization which we have found to be very natural for this problem has been named *the inductive parameterization*. Assuming the initial primary segment $\mathbf{U}_0 = \{\mathbf{U}_0(t)|0 \leq t \leq 1\}$ has been given some parameterization (which is discussed below), let $\mathbf{U}_1(t) = f(\mathbf{U}_0(t-1))$ for $1 \leq t \leq 2$. Then by mathematical induction, we can define $\mathbf{U}_n(t)$ in terms of $\mathbf{U}_{\mathbf{n-1}}(t-1)$ and thus parameterize the whole curve.

We find the following example of the inductive parameterization both instructive and useful in the problem of computing unstable manifolds. Consider a linear map in the form $\mathbf{x}' = A\mathbf{x}$ with a hyperbolic fixed point at the origin. Let λ be the unstable eigenvalue and $\mathbf{v}$ be its associated eigenvector, and let $\mathbf{x}_0 = c\mathbf{v}$, W^{u} is of course equal to span$\{\mathbf{v}\}$. The initial primary segment of the unstable manifold can be written as $\mathbf{U}_0 = W^{\mathrm{u}}[\mathbf{x}_0, \lambda\mathbf{x}_0]$. If this segment is parameterized by $\mathbf{U}_0(t) = \lambda^t x_0$ for $0 \leq t \leq 1$, then using the algorithm described above extends this parameterization inductively to all segments $\mathbf{U}_n$ and thus to the whole branch of W^{u}.

This example suggests a way to parameterize the initial primary segment $\mathbf{U}_0$. Near the fixed point (which we can assume is $\mathbf{x}^* = \mathbf{0}$), the map $f(\mathbf{x})$ is approximately given by $\mathbf{x}' = Df(\mathbf{0})\mathbf{x}$ so we can approximate $\mathbf{U}_0$ by an appropriate segment of the unstable eigenspace using the above parameterization. We discuss a nonlinear correction to this approach in Sec. 7.

Another motivation for the inductive parameterization is given by the "parameterization method," a rigorous analytical method of computing invariant manifolds of hyperbolic fixed points, e.g. [Cabré *et al.*, 2003a, 2003b, 2005; Franceschini & Russo, 1981]. The parameterization method is based on power series expansion. The stable or unstable manifolds of map f at a fixed point $\mathbf{x}_0$ is represented by a parametric function $P(s)$ which is characterized by the invariance equation

$$(f \circ P)(s) = P(\lambda s), \tag{3}$$

where λ is a stable or an unstable eigenvalue. Substituting $s = \lambda^t$ in (3) we obtain $(f \circ P)(\lambda^t) = P(\lambda^{t+1})$ which for any primary segment P is equivalent to our inductive parameterization. The parameterization method is the basis for the recent computations [Mireles-James, 2009; Mireles-James & Lomelí, 2010].

4. Brief Introduction to Geometric Modeling Tools

For the sake of the flow of the paper, we give here a very brief sketch of the geometric modeling tools from which our algorithm is constructed. A more detailed introduction is given in Appendix A, which we encourage for further reading. We mention here only ideas that may be unfamiliar to specialists in dynamical systems, although in the appendix some more familiar ideas are discussed in the context of CAGD.

Bézier Curves refer to a family of degree-n curves defined in terms of $n+1$ *control points* $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$, with $n = 3$ being the most common choice, for a parameter interval $[a, b]$. These curves interpolate the first and last points $\mathbf{p}_0$ and $\mathbf{p}_n$ at parameter values a and b, respectively, while the other points merely influence the shape of the curve. An important property of these curves is the *convex hull property*, namely that the curve lies inside the convex hull of the control points for all parameter values between a and b. We make use of this property to define the adaptivity condition.

In many problems, one would like to construct a curve that interpolates (or comes very close to) the points $\{\mathbf{x}_0, \ldots, \mathbf{x}_n\}$ at the parameter values $\{t_0, \ldots, t_n\}$. A degree-n Bézier curve does not solve this problem, so we turn to piecewise Bézier curves, namely *Catmull–Rom splines*. These satisfy certain continuity properties, while maintaining *local control*, i.e. a change to a given point $\mathbf{x}_k$ will change the curve only on some restricted parameter interval $[t_{k-j}, t_{k+j}]$ rather than on the whole interval $[t_0, t_n]$. Recall that a *cubic Hermite interpolant* is defined to match both the function values and the tangent vectors at the endpoints of a curve segment. A Catmull–Rom spline is defined to be an Hermite curve, where the tangent vectors are determined by finite-difference approximations of the interpolation data. We will use both third-degree and fifth-degree finite differences to approximate these tangent vectors.

In Appendix A, all these families of curves are precisely defined, and a fuller explanation is given of their desirable properties, and the relations between them. We also discuss various notions of continuity, efficient algorithms for evaluating these curves, and the expected accuracy of the various approximations made.

5. Adaptive Methods

In the standard interpolation problem of CAGD, as stated for example [Farin, 2002], one is given a sequence of points $\mathbf{x}_0, \ldots, \mathbf{x}_n$ and would like to draw a curve passing through those points in the given order. Our goal is somewhat different. We seek to choose the points $\mathbf{x}_k = \gamma(t_k)$ as efficiently as possible in order to accurately render the curve.

5.1. *Existing methods*

Algorithms for efficiently computing unstable manifolds go back at least to the 1980's [Parker & Chua, 1989]. Two methods developed for drawing unstable manifolds of maps are due to [Hobson, 1993] and [Carter, 2004]. Both are essentially algorithms for computing a parametric curve γ in Eq. (2), although neither is framed this way. Suppose a model curve is defined by piecewise linear interpolation between $n+1$ points $\{\mathbf{x}_0 = f(a), \mathbf{x}_1 = f(t_1), \ldots, \mathbf{x}_n = f(b)\}$. Define the lengths $l_k = \|\mathbf{x}_k - \mathbf{x}_{k+1}\|$ and the angles α_k between consecutive linear segments $\mathbf{x}_{k-1}\mathbf{x}_k$ and $\mathbf{x}_k\mathbf{x}_{k+1}$. Then in both methods, a model curve is considered acceptable if it satisfies

$$\alpha_k < \mathbf{tol}_1, \tag{4a}$$

$$\alpha_k l_k < \mathbf{tol}_2, \quad \text{and} \quad \alpha_k l_{k-1} < \mathbf{tol}_2, \tag{4b}$$

where $\mathbf{tol}_1$ and $\mathbf{tol}_2$ are user-specified tolerances. The first condition (4a) states that two consecutive segments should be in nearly the same direction, while the last two [Eq. (4b)] help to control the arc length. The first condition is scale-invariant while the second has an absolute scale. A schematic of such a linear interpolant, together with the associated angles and lengths, is shown in Fig. 3.

5.1.1. *Hobson's method*: *Marching*

Hobson's method begins at $t = a$ and adds new points $\mathbf{x}(t_k)$ to a given list of points until $t = b$ is reached. The method supposes that a sequence of points $\mathbf{x}_k = f(t_k)$ has already been found and attempts to find the next point $\mathbf{x}_{k+1} = f(t_{k+1})$. Let s be a short parameter increment and let $t' = t_k + s$, $t'' = t_k + 2s$, $\mathbf{x}' = f(t')$ and $\mathbf{x}'' = f(t'')$. Define the vectors $\mathbf{v}_1 = \mathbf{x}_k\mathbf{x}'$ and $\mathbf{v}_2 = \mathbf{x}'\mathbf{x}''$. Let $d = \|\mathbf{v}_1\|$ and α be the angle between $\mathbf{v}_1$ and $\mathbf{v}_2$. Hobson's algorithm uses the conditions

$$\alpha < \mathbf{tol}_1, \tag{5a}$$

$$\alpha d < \mathbf{tol}_2 \tag{5b}$$

to determine whether to accept the point $\mathbf{x}'$. If the tolerances are not met, the algorithm decreases s and then tries again. In either case, the point $\mathbf{x}''$ is discarded. No explicit mention is made in [Hobson, 1993] on the condition for choosing s, although the method for choosing how to adjust s can clearly make a large difference in the efficiency of the algorithm. For example, one might take

$$s \to 0.95 \max\left\{\frac{\mathbf{tol}_1}{\alpha}, \frac{\mathbf{tol}_2}{\alpha d}\right\} s$$

after each step, regardless of whether the current step is accepted.

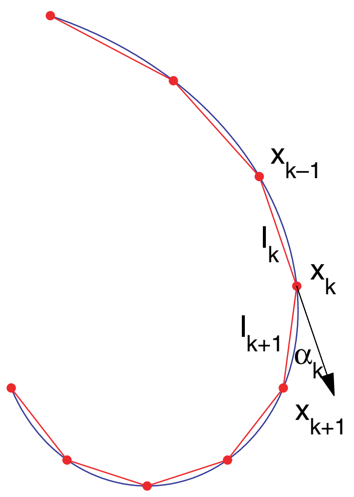


Fig. 3. The angle between consecutive segments in a discrete curve.

There are two sources of inefficiency in this method. First, at each step, the computation of $\mathbf{x}''$ does not contribute to a point on the curve. In addition, each time a point $\mathbf{x}'$ is rejected, two points are computed that do not contribute to the curve. Further note that the conditions (5) for accepting $\mathbf{x}'$ are not equivalent to the conditions (4). It is possible, for the algorithm to satisfy conditions (5) while badly failing to satisfy conditions (4); see Fig. 4. An obvious downside to this method is the need to compute and discard a point $\mathbf{x}''$ at each step. A simpler method, given the curve up to $\mathbf{x}_k$, would be to compute a candidate point $\mathbf{x}'$ and test whether the three points $\mathbf{x}_{k-1}, \mathbf{x}_k$ and $\mathbf{x}'$ satisfy conditions (4). Hobson demonstrates that this method can fail at instances that $\mathbf{x}_k$ itself is too far from $\mathbf{x}_{k-1}$ for any such $\mathbf{x}'$ to produce an acceptable value of the angle α, in particular if $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ are situated on opposite sides of a hairpin turn in curve γ, as in Fig. 4.

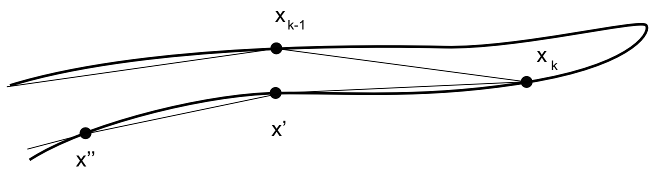


Fig. 4. A situation where part of the curve might be missed.

This method is widely cited, as is a similar method [Krauskopf & Osinga, 1998b].

5.1.2. *Carter's method*: *Bisection*

Carter's method, by contrast, is based on bisection. Given an approximation to the whole curve γ, the algorithm calculates all the angles α_k between the segments as shown schematically in Fig. 5. For each point x_k at which either condition of (4) is violated, the algorithm adds new points at $t_k^+ = (t_k + t_{k+1})/2$ and $t_k^- = (t_{k-1} + t_k)/2$. A new approximate curve is constructed interpolating $\{\mathbf{x}(t_k)\} \cup \{\mathbf{x}(t_k^{\pm})\}$.

Carter compares his algorithm to Hobson's and finds that for strict tolerances, his method can draw the curve with as few as one third as many calls that to the function f; see Table 1. He found in practice, it was sufficient to enforce only condition (4a) while ignoring (4b). Our experience is similar. This method has appeared only as a preprint and as an undergraduate honors thesis, and so has not been cited, although it is very similar to a method used, without description, to draw the tangles in [Goodman *et al.*, 2002, 2004]. Although Carter was the first to propose this method in the context of invariant manifold calculations, it was earlier discussed in the computer graphics literature for parametric curves, although these papers are not well-cited. The method was proposed in [Chandler, 1990] and subsequently in [Tzafestas & Pantazopoulos, 1999] and [de Figueiredo, 1995]. In these cases, the authors' focus is on the rendering of parametric curves on computer screens or printers. Thus the refinement condition for the recursion is related to ensuring that any errors be smaller than one pixel. For dynamics problems, we may be interested in resolving features of the curve that are finer than what is visible on the screen or on paper. We thus subject these methods to some exacting numerical diagnostics to test their effectiveness. In what follows, we refer to Carter's method as **A**daptive **L**inear **I**nterpolation or **ALI**.

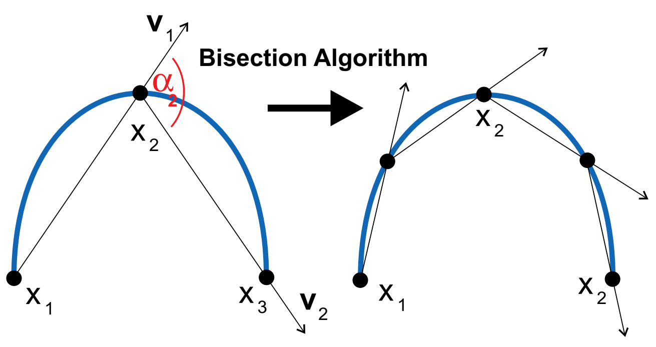


Fig. 5. The angle between consecutive linear segments and the bisection algorithm.

5.2. *The proposed methods*

We will demonstrate that ALI, while superior to Hobson's method, does not compute invariant manifolds efficiently because it places too many points near hairpin turns. We propose to remedy this by using the Catmull–Rom spines described briefly in Sec. 4 and more thoroughly in Appendix A. One familiar way to interpolate n of points is with cubic splines — piecewise cubic curves chosen to satisfy matching conditions at the interpolated points so that the overall curve is C^2. This scheme has two disadvantages which have led to its rejection in CAD and computer animation systems. First, satisfying the matching conditions requires solving a linear system, so that if new points are added, the entire object must be recomputed. Second, and closely related, the method is nonlocal; changes to one interpolated point lead to changes in all the coefficients. Interpolation methods used in computer animation generally use piecewise-cubic splines, but abandon the C^2-matching requirement, which allows for far greater flexibility. Appendix A describes various equivalent forms for piecewise-polynomial curves, and it is convenient to use different forms for different aspects of a calculation.

5.2.1. *Flatness refinement condition*

We now discuss what the refinement condition, the analog to condition (4a), should be for a recursive method based on Catmull–Rom splines. The Catmull–Rom spline is by its definition C^1 at the interpolated points, so that the angle α between two consecutive segments is identically zero. Because of this, our method depends on a condition applied *within* a single segment as opposed to the *between-segments* condition (4a). Geometrically, we require that each segment be sufficiently flat.

Several possible flatness conditions are suggested by the equivalent Bézier form of the Catmull–Rom interpolant. Referring to the schematic of a Bézier control polygon given in Fig. 6, several possible measurements of the flatness of the Bézier curve are given by

(1) $\max\{d_1, d_2\}$ (distance from interpolating line to the further control point);

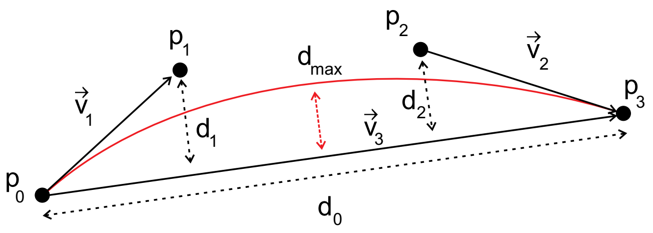


Fig. 6. The control points and several characteristic quantities of a Bézier curve.

(2) $(|\mathbf{p}_0\mathbf{p}_1| + |\mathbf{p}_1\mathbf{p}_2| + |\mathbf{p}_2\mathbf{p}_3| - |\mathbf{p}_0\mathbf{p}_3|)$;
(3) $\max\{d_1/d_0, d_2/d_0\}$ (aspect ratio of the bounding box);
(4) the angle between the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$;
(5) $(|\mathbf{p}_0\mathbf{p}_1| + |\mathbf{p}_1\mathbf{p}_2| + |\mathbf{p}_2\mathbf{p}_3| - |\mathbf{p}_0\mathbf{p}_3|)/|\mathbf{p}_0\mathbf{p}_3|$.

Each of the above conditions introduces a different approach to estimate the flatness. Moreover, conditions 1 and 2 have units of length whereas conditions 3–5 are dimensionless. These conditions might seem arbitrary but are closely related to the geometry of Bézier curves. For example, the convex hull property guarantees that $d_{\max}$, the maximum distance between $\gamma(t)$ and its interpolant is bounded by condition 1.[2]

To motivate the flatness refinement condition we look at an application of the de Casteljau algorithm in Appendix A, which is often used to adaptively render Bézier curves. The adaptive algorithm works by splitting a given cubic Bézier curve defined for $t \in [a, b]$ into two equivalent Bézier curves, defined on the intervals $[a, (a + b)/2]$ and $[(a + b)/2, b]$, with the algorithm producing the value of the Bézier curve at the midpoint [choosing $t = (a + b)/2$ in Eq. (A.6)]. The algorithm chooses whether to split the interval based on a flatness condition like those on this list, often based on the resolution of the display device.

Catmull–Rom splines based on three-point centered differences or five-point centered differences, along with one of the above flatness refinement conditions, give our next methods for adaptively resolving a parametric curve. We refer to these methods as **A**daptive **C**atmull–**R**om 3, (**ACR3**) and **A**daptive **C**atmull–**R**om 5, (**ACR5**), respectively.

5.2.2. *Error refinement condition*

We find in the numerical tests below that a slightly improved version of ALI performs nearly as well as both these ACR methods. We note that each of these schemes features a refinement condition based on properties of a single model curve. Adaptive methods for other types of problems, for example, adaptive quadratures and adaptive ODE solvers, work by comparing two approximations in order to estimate an error and then refining near the locations where they disagree. We propose such a scheme here. At each iteration we compute both the 3-point and 5-point Catmull–Rom splines described in Appendix A. The difference between them will give the refinement condition. As it is expensive to find the actual distance between two Bézier curves, we estimate it using their control polygons. To estimate the distance between two Catmull–Rom approximations we use the equivalent Bézier form of the Catmull–Rom interpolant. We refer to the schematic of a Bézier curve; see Fig. 7.

This schematic depicts the difference between two different interpolants to the same sequence of points. Since the two curves interpolate the same data, the first and the last points of the difference curve are zero. Several possible estimations of the error between the two approximations are given by

(1) $\max\{|\mathbf{w}_1|, |\mathbf{w}_2|\}$; (maximum distance between the control points)
(2) $\max\{|\mathbf{w}_1|/d_0, |\mathbf{w}_2|/d_0\}$; (maximum relative distance between the control points).

Condition 1 has units of length whereas condition 2 is dimensionless. However, both these approximations are computationally less expensive than computing the maximum norm error directly; we can avoid "calculation" of the curve and estimate the error using the already-calculated control points.

The adaptive method for rendering a parametric curve based on one of these error refinement conditions we call 3-point versus 5-point **A**daptive **C**atmull–**R**om or **ACR3vs5**.

5.3. *Other methods*

Similar methods for computing the unstable manifold of planar maps are developed in [England *et al.*, 2005; Krauskopf & Osinga, 1998a, 1998b, 1999; Krauskopf *et al.*, 2004]. These are based on similar conditions to Eqs. (5). The manifold is computed by adding one point after the other without using the idea of primary segments. However, the method is still based on linear interpolation.

[2]In fact, an elementary calculus exercise demonstrates that $d_{\max} \leq (3/4)\max\{d_1, d_2\}$, see [Cheng, 1992].

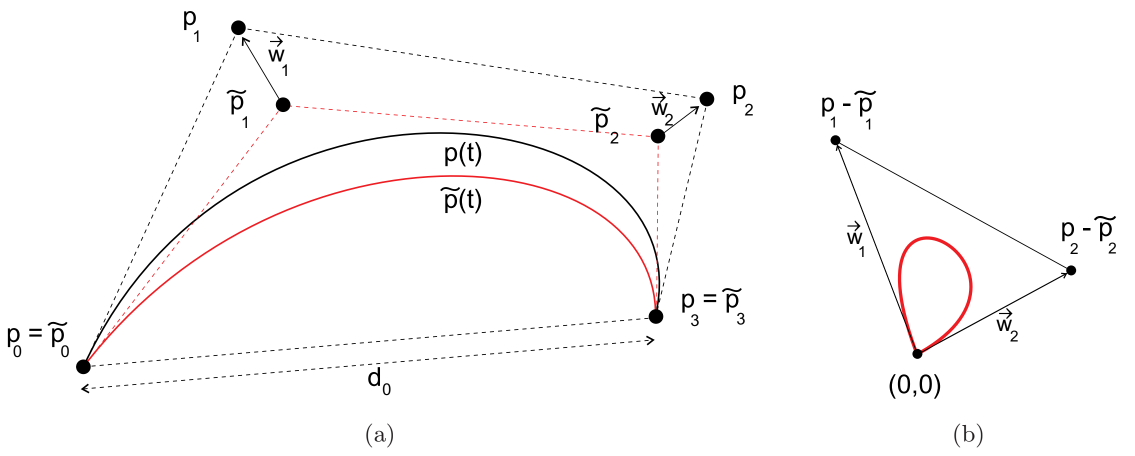


Fig. 7. Two different Catmull–Rom approximations, $\mathbf{p}(t)$ and $\tilde{\mathbf{p}}(t)$, of the curve $\gamma(t)$. (a) The two approximations $\mathbf{p}$ and $\tilde{\mathbf{p}}$. (b) The error loop, $\mathbf{p} - \tilde{\mathbf{p}}$.

You *et al.* [1991] presented a slightly different approach. They do use the idea of primary segments. Their adaptive method is however, controlled by the distance between resulting points. The procedure does not require an interpolation of any primary segment except the initial one, as long as the distance is maintained below a screen resolution.

Additionally, there exists an important rigorous, analytical method of computing invariant manifolds based on power series expansions. This method has been used both to prove analytical statements about invariant manifolds [Cabré *et al.*, 2003a, 2003b, 2005] and to numerically calculate such manifolds [Franceschini & Russo, 1981]. In this method, a branch of the unstable manifold is written as $\{(x(s), y(s)) \,|\, 0 < s < \infty\}$ where x and y are represented as power series in s, and the invariance of the manifold under the map is used to derive equations for the coefficients in the series. If f is entire, this series has infinite radius of convergence, but, in practice, due to roundoff error, the numerical radius of convergence may be quite short, although there exist methods to increase this radius (without resorting to variable precision arithmetic). The method is also the basis for the recent computations of two-dimensional invariant manifolds, and their intersection in [Mireles-James, 2009; Mireles-James & Lomelí, 2010].

We mention this method briefly in Sec. 3 as the parameterization of the power series is closely related to our "inductive parameterization." The method eliminates the need to compute the images of fundamental segments, and, thus, the propagation of error due to interpolation. Nonetheless, this method does not invalidate the work presented here. While this method provides an explicit formula to compute points on the manifold, it thus reduces the problem to that of drawing a parametric curve, i.e. choosing which points to plot and how to interpolate between them, which is exactly what our methods provide. Further, in the case that the map is given as a numerical routine and not by an analytical formula (e.g. as a numerical Poincaré map), this method is not applicable. Finally when the map is given by an equation with complex singularities (e.g. poles) the power series may converge very slowly, thus giving a poor approximation.

6. Numerical Tests of Proposed Tools

In Sec. 8 we will test the algorithms using invariant manifolds, but by performing our initial numerical experiments on explicit parametric curves, we can gain a bit more control over the testing process and gain a clearer understanding of the behaviors of the methods.

We introduce a model curve with which we test our algorithms, given in polar coordinates as $r = r(t)$ and $\theta = \theta(t)$:

$$\gamma_{\text{test}} = \Big\{(r,\theta) : r = 1 + \epsilon(3t + \cos t + \cos\sqrt{2}t), \\ \theta = \frac{\pi}{2}(\sin t + \sin\sqrt{2}t), a \le t \le b\Big\}, \tag{6}$$

and shown in Fig. 8.

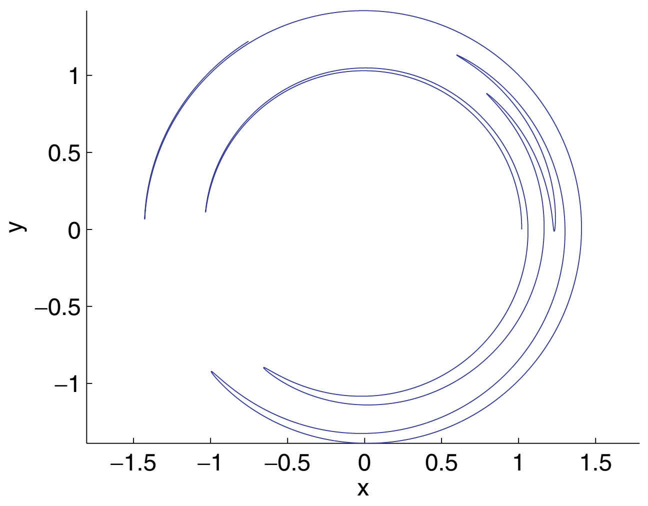


Fig. 8. The test curve for $\epsilon = 10^{-2}$, $0 \le t \le 15$.

The curve has portions that are nearly circular, connected by regions where the curve makes a sharp turn with large curvature. The curvature of the test curve may be increased as desired by adjusting the parameter $\epsilon \ll 1$. We consider the curve a "model" of unstable manifolds in the sense that it has regions of both large and small curvatures, alternating somewhat unpredictably, and is nonself-intersecting. Since the nearly circular portions of this curve lie very close together, it can be used to test modifications to the algorithm to prevent self-intersection of the approximate curve.

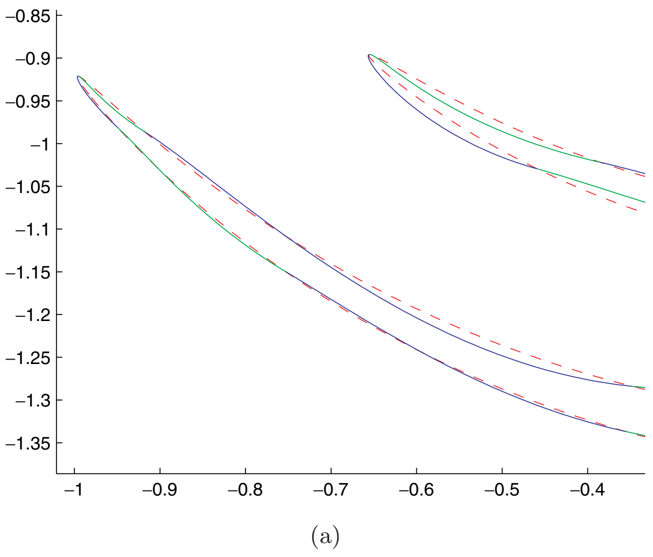
6.1. *Numerical test of ACR3*

We start with a visual test of the flatness refinement condition with the Catmull–Rom spline based on three-point central differences on the test curve γ_{test}, (6). We tested all of the presented flatness conditions and decided to use condition 1 for further work. For various values of the flatness tolerance we visually checked the graphs containing both the exact and approximate curves to judge how "close" the interpolated curve is to the exact curve. In our initial explorations, we noticed that the adaptive Catmull–Rom spline approximation performs well along the curve except near sharp tips, where the approximation has undesirable wiggles; see Fig. 9(a). The observed "wiggles" correspond to unwanted variation in the curvature of the C^1 Catmull–Rom spline and suggest an improvement. An improved method, which we call ACR3+, has two steps

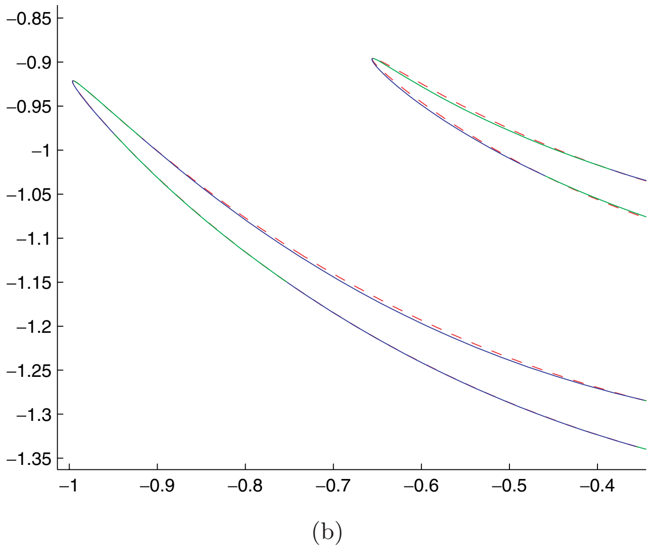
(1) use the ACR3 method to generate the points along the curve,
(2) take the model curve to be the natural cubic spline[3] interpolating these points.

Note the improvement in Fig. 9(b).

In this section, we describe the initial numerical tests performed to demonstrate the superiority of the ACR3 and ACR3+ methods over ALI. Additionally, to make a fair comparison between



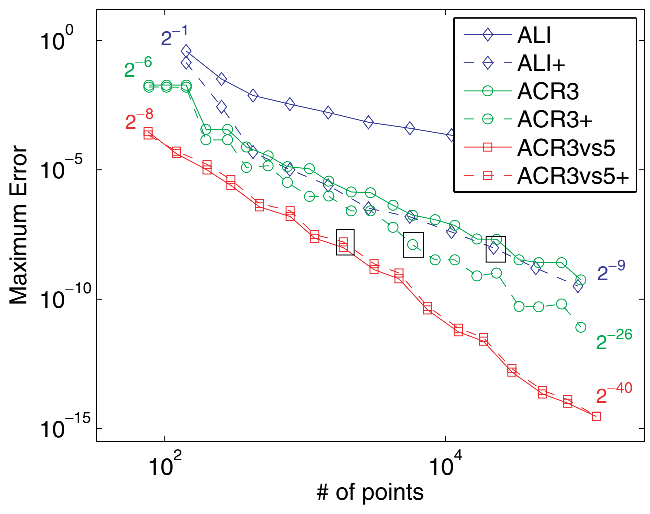
(a)



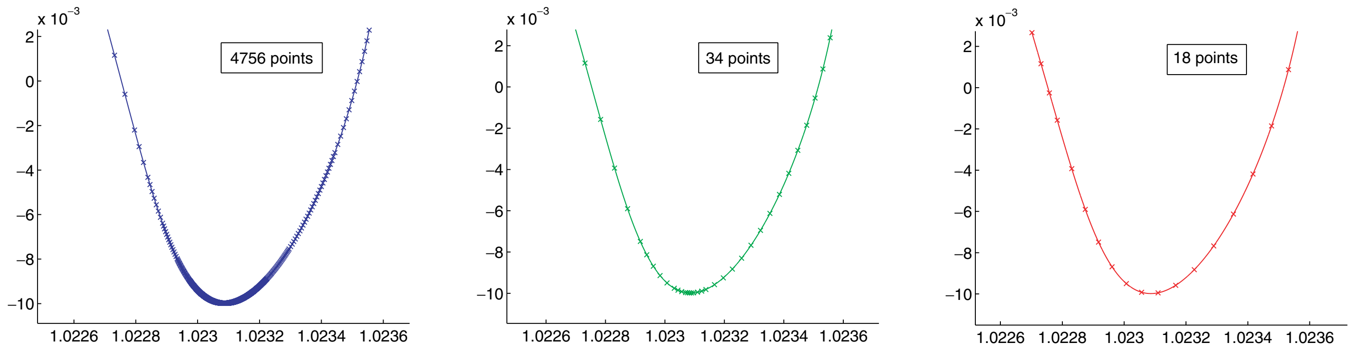
(b)

Fig. 9. The exact curve (red dashed) and two different spline interpolants on the knots from the adaptive Catmull–Rom method, part of the γ_{test} curve with $\epsilon = 10^{-2}$. (a) ACR3. (b) ACR3+.

[3]In all numerical tests we use MATLAB's built-in `spline` command by default, which uses the not-a-knot condition at the endpoints and is more accurate than the natural spline. For convenience, we will refer to this as a natural cubic spline.



(a)



(b) (c) (d)

Fig. 10. (a) Convergence of the error for the adaptive methods with different refinement conditions changing as indicated, between the values at the opposite ends of each graph. The methods tested on the test curve with $0 \leq t \leq 10$ and $\epsilon = 10^{-3}$. Comparison of (b) the ALI+ method, (c) the ACR3+ method and (d) the ACR3vs5 method on one sharply pointed segment of the curve for $t \in (7.57, 7.82)$, the error approximately at the same 10^{-8} level indicated by boxes in (a). The curvature at this point is about 1.1×10^5.

them, we introduce one more method. The procedure defined by first using the ALI method to generate points and then passing a natural cubic spline through those points we call ALI+. In Fig. 10(a), we plot the maximum error $\varepsilon = \max_{t\in[0,10]} |\gamma_{\text{exact}}(t) - \gamma_{\text{approx}}(t)|$ as a function of the number of points used in each computation as the tolerance is decreased. The error is computed at one hundred points between every two interpolated points and the approximate maximum taken as the maximum over this finite set.

First, it is clear that ALI+ converges much faster than ALI and that ACR3+ provides a more modest improvement over ACR3. Our discussion of Fig. 9 shows that although this improvement might be rather small quantitatively, it can provide significant qualitative improvement. A common technique used in evaluating CAGD methods is to plot the curvature as a function of the parameter; see [Farin, 2002, Chapter 9]. A "good" method would be one in which the computed curvature does not contain unnecessary oscillations. Although we do not provide such a plot here, Fig. 9 makes it clear that ACR3+ would be seen to be better in this case.

It is also clear from Fig. 10 that ACR3 converges much more rapidly than ALI, but that the advantage of ACR3+ over ALI+ is far less dramatic. Nonetheless, the improvement is significant. In Fig. 10(a), we have drawn small boxes over the points corresponding to the largest tolerance for which the maximum error falls below 10^{-8}, for the ALI+ and ACR3+ (and ACR3vs5) methods. The computation using ALI+ required 22 125 total points while that using ACR3+ required only 5838

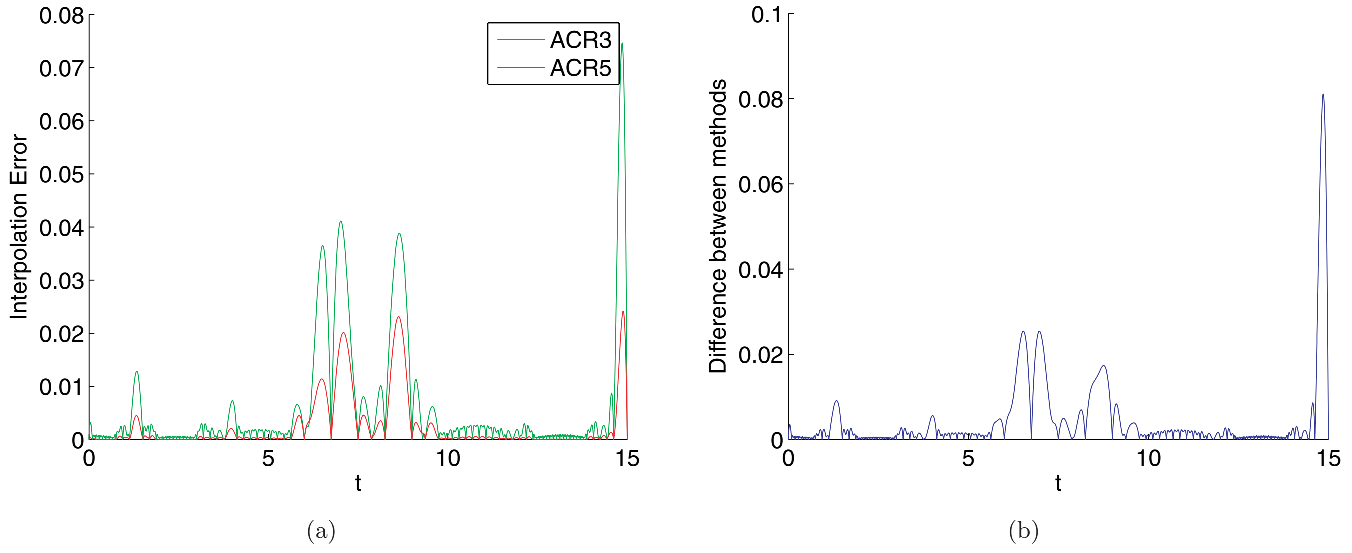


Fig. 11. Test on the model curve (6) with $\epsilon = 10^{-2}$, $0 \le t \le 15$, (a) the error of the Adaptive Catmull–Rom methods, ACR3 green and ACR5 red, (b) the difference between the ACR3 and ACR5 approximations. (a) Flatness condition at the level 2^{-4}. (b) Difference between approximations.

points. To illustrate the origin of this savings, we show in Figs. 10(b) and 10(c) the points computed by the two methods in the neighborhood of one hairpin turn. ALI+ has required 4756 points in this neighborhood, to 34 for ACR3+.

6.2. *Numerical test of ACR3vs5*

Figure 10(a) gives a convenient summary of the performance of the various methods, but does not allow us to understand the cause of the disappointing performance. For this we must understand where large errors occur. Figure 11(a) shows the local error $|\varepsilon(t)|$ in both the ACR3 (Catmull–Rom with three-point centered differences) and ACR5 (Catmull–Rom with five-point centered differences) methods after one run with fairly large tolerance. While both curves satisfy the flatness condition for all values of t, the error in the approximation varies widely. Comparing Figs. 11(a) and 11(b) we see that the difference between the exact curve and each of the approximations is large exactly for values of t where the two approximations disagree with each other. These figures allow us to both diagnose and correct the problem with the ACR3 method. The flatness condition measures a property of the interpolating curve itself, one that clearly is not well-correlated with the interpolation error. The difference between the two approximations does correlate well with error and makes a more effective refinement condition. This motivates the ACR3vs5 method introduced in Sec. 5.2.

After experimenting with both the error conditions discussed in Sec. 5.2, we chose to work with Condition 1. We call the method combining **ACR3vs5** with a natural cubic splines **ACR3vs5+**. We test both methods ACR3vs5 and ACR3vs5+ on the test curve (6) along with the previous method ALI, ALI+ and ACR3 and ACR3+.

The final two lines in Fig. 10 show the convergence of the two error-refinement methods. The method ACR3vs5 converges faster than ALI, ALI+, ACR3 and ACR3+. This improvement leads us to expect that applying method ACR3vs5 to computing invariant manifolds should lead to similar results. Note that ACR3vs5+ performs no better than ACR3vs5.

7. Details of the Numerical Implementation Invariant Manifold Calculation

7.1. *Some notation*

Let U_n be the numerical approximation to the nth true segment $\mathbf{U}_n$. This approximation is defined as the Catmull–Rom curve interpolating the points in the set

$$\mathcal{X}_n = \{\mathbf{x}_k^n\}_{k=0}^{N_n}$$

at the corresponding parameter values in the set

$$\mathcal{T}_n = \{t_k^n\}_{k=0}^{N_n}.$$

In particular, we can define the operator Γ_{CR} that maps a set of parameter values and the associated interpolation points to their Catmull–Rom interpolant, i.e.

$$U_n = \Gamma_{\mathrm{CR}}(\mathcal{T}_n, \mathcal{X}_n).$$

Thus we may construct the approximate invariant manifold using an inductive procedure, with two obvious steps. First, to construct the initial segment $U_0 = \Gamma_{\mathrm{CR}}(\mathcal{T}_0, \mathcal{X}_0)$, which must satisfy the chosen refinement conditions. Second, given U_n, to find suitable sets $\mathcal{T}_{n+1}$ and $\mathcal{X}_{n+1}$, which define an approximation $U_{n+1} = \Gamma_{\mathrm{CR}}(\mathcal{T}_{n+1}, \mathcal{X}_{n+1})$ which satisfies the same refinement condition. In order to make this construction, we introduce two new pieces of notation:

$$\mathcal{T}_n + 1 = \{t_k^n + 1\}_{k=0}^{N_n} \quad \text{and} \quad f(\mathcal{X}_n) = \{f(\mathbf{x}_k^n)\}_{k=0}^{N_n}.$$

We call a segment *resolved* if it satisfies adaptive refinement condition being used for the particular algorithm.

Consecutive primary segments are joined together so that the last point in U_n is the first point in U_{n+1}, i.e. $\mathbf{x}_0^n = f^n(\mathbf{x}_0^0)$. Subsequent primary segments can be found in the same way recursively. According to the inductive parameterization from Sec. 3, all values $t_k^n \in \mathcal{T}_n$ are drawn from the interval $[n, n+1]$.

7.2. *The initial primary segment*

In order to initialize any adaptive method for computing an invariant manifold, one first has to determine the initial primary segment U_0, approximating the true segment $\mathbf{U}_0$, as described in Sec. 2. The easiest and the most commonly used method is to choose the initial primary segment as a line segment from the unstable subspace. It is crucial to determine U_0 very accurately in order to avoid error propagation in further computations. Therefore, this line segment must be taken very close to the fixed point. This approach is not practical when we are interested in computing a fairly long portion of the manifold. Setting the initial primary segment closer to the fixed point requires many more iterations to compute the same portion of the manifold. In order to be more efficient, the initial primary segment should be taken further from the fixed point, using higher-order approximation. Accordingly, U_0 is defined as a polynomial expansion of the manifold close to the hyperbolic fixed point.

We consider the map of the form $\mathbf{x}' = f(\mathbf{x})$ as described in Sec. 2, where $\mathbf{x} = (x, y)$, $f : \mathbb{R}^2 \to \mathbb{R}^2$ and the prime denotes the forward mapping. The map can be written as

$$x' = f_1(x, y)$$
$$y' = f_2(x, y).$$

Near the hyperbolic fixed point (x^*, y^*) a branch of invariant manifold can be explicitly written as $y = p(x) = \sum_{k=0}^{\infty} c_k (x - x^*)^k$. The invariance of the manifold under the map f, yields the algebraic relation

$$f_2(x, p(x)) = p(f_1(x, p(x))). \tag{7}$$

Solving this recurrence yields $c_0 = y^*$ and two possible values for c_1, corresponding to the stable, W^{s} and the unstable, W^{u} manifold. Once c_1 is chosen, the remaining coefficients can be found uniquely in ascending order from relation 7. The single branch of the manifold near the hyperbolic fixed point (x^*, y^*) can be approximated by $y = p_N(x)$, the truncation of the series to the $(x - x^*)^N$ term. Taking $(x_0 - x^*)$ small enough and N large enough that

$$|f_2(x_0, p_N(x_0)) - p_N(f_1(x_0, p_N(x_0)))| < \varepsilon \ll 1,$$

guarantees that the error between the polynomial approximation U_0 and the true manifold $\mathbf{U}_0$ along this segment is less than δ. The same approach is used in [Hobson, 1993]. Moreover, our further numerical tests (not shown here) show that this is necessary in order to avoid error propagation, and, especially, to smoothly join two segments U_n and U_{n+1}.

We want to generate the sets $\mathcal{X}_0$ and $\mathcal{T}_0$ in order to construct an accurate approximation U_0. Choosing x_0^0, the x-coordinate of the first point of the initial segment, the y-coordinate y_0^0 is determined by the expansion $p(x)$ as $y_0^0 = p(x_0^0)$. According to the definition of U_0 the last point of the segment is determined by the choice of the first point and its image $(x_{N_0}^0, y_{N_0}^0) = f(x_0^0, y_0^0)$.

Having the first and the last points of the initial primary segment we generate the set of intermediate points. Let I_0 be the interval of the x-coordinates between x_0^0 and $x_{N_0}^0$. This is parameterized by the inductive parameterization introduced in Sec. 3 where the eigenvalue of $Df(\mathbf{0})$ is replaced by a nonlinear correction $\lambda = (x_{N_0}^0 - x^*)/(x_0^0 - x^*)$. Then any point of the U_0 has the x-coordinate in a form $x_i^0 = x^* + \lambda^{t_i}(x_0^0 - x^*)$ where $t_i \in [0, 1]$ is a parameter value. The y-coordinate of the points are given by the manifold expansion as $y_i^0 = p(x_i^0)$. The complete set of points with their parameter values determines a discrete representation of the initial primary segment. In this way we define the sets $\mathcal{T}_0$ and $\mathcal{X}_0$, which give $U_0 = \Gamma_{\rm CR}(\mathcal{T}_0, \mathcal{X}_0)$. In order to obtain an accurate interpolant at later steps, we found it necessary to use a large number of points, about 50.

7.3. *Resolving a simple primary segment*

We restrict our attention to the specific problem of finding the unknown primary segment U_{n+1} which is the approximate image under the map f of an already-resolved primary segment $U_n = \Gamma_{\rm CR}(\mathcal{T}_n, \mathcal{X}_n)$, i.e. $U_{n+1} \approx f(U_n)$. Mapping the segment U_n forward yields a parametric curve $\tilde{U}_{n+1} = \Gamma_{\rm CR}(\mathcal{T}_n + 1, f(\mathcal{X}_n))$ which approximates the next primary segment U_{n+1}. As the dynamics is expanding along the manifold, we expect that a larger set of points should be needed to resolve the segment U_{n+1} than was required for U_n. The curve $\tilde{U}_{n+1}$ will in general be unresolved, but will indicate the general shape of U_{n+1}.

Having already generated the initial guess $\tilde{U}_{n+1}$ we may apply any of the proposed adaptive method from Sec. 5.2 to produce a resolved approximation $U_{n+1}(t) = \Gamma_{\rm CR}(\mathcal{T}_{n+1}, \mathcal{X}_{n+1})$ to the desired parametric curve $f(U_n)$. Note that by construction, the number of points on consecutive segments satisfy $N_{n+1} \geq N_n$.

7.4. *Kink patching*

The proposed methods are quite capable of resolving a single primary segment of a manifold given a previously completed segment. Segments U_n and U_{n+1} may be individually resolved but the composite curve formed by their union $U_n \cup U_{n+1}$ may be unresolved in the neighborhood of their common point. We call a point where this occurs a kink.[4] Therefore, some care must be taken to ensure that the composite curve formed by two successive primary segments is also well-resolved around that point. This can be accomplished by examining a small set of consecutive points centered about the joint point.

Consider the kink between already-resolved segments U_n and U_{n+1}. The points before the joint are taken from the end of the discrete representation of U_n and require preimages in $U_{n-1}(t)$, the points after are from the beginning of U_{n+1} and require preimages in $U_n(t)$. The inductive parameterization presents a clear advantage in searching for preimages.

Assume that a subsegment of U_n has to be refined between points $\mathbf{x}_i^n$ and $\mathbf{x}_{i+1}^n$. The parameter values corresponding to that pair are t_i^n and t_{i+1}^n. Their preimages can be found by evaluating the previous segment approximation $U_{n-1}(t)$ at $t_i^n - 1$ and $t_{i+1}^n - 1$. In order to refine the subsegment of U_n it is sufficient to sample the approximation $U_{n-1}(t)$ on $t_* = (t_i^n + t_{i+1}^n)/2 - 1$ and map it forward, $\mathbf{x}_*^n = f(U_{n-1}(t_*))$. The new point lies on U_n between points $\mathbf{x}_i^n$ and $\mathbf{x}_{i+1}^n$, the new parameter value for this point is $t_*^n = t_* + 1$.

When the joint kink between segments U_n and U_{n+1} is resolved, then we can say that the segment U_n is completed. Even though the segment U_{n+1} is well resolved it may be necessary to refine it again close to the joint vertex with a segment U_{n+2}.

The procedure is only slightly different for patching the kink between segments U_0 and U_1. To refine the end of U_0 we just take more points from $U_0(t)$, the continuous approximation of the initial primary segment.

8. Numerical Tests

Example 1. *Hénon Map*

One of the most studied examples of dynamical systems that exhibit a chaotic behavior, this map is given by

$$x' = 1 + y - ax^2;$$
$$y' = bx.$$

The closure of one branch of the unstable manifold is a well-studied strange attractor with fractal structure. We have chosen this example primarily for the purpose of visualization.

For the standard parameters $a = 1.4$ and $b = 0.3$ the map has a hyperbolic fixed point at

$$(x^*, y^*) = (-1.131354477089505, -0.339406343126851).$$

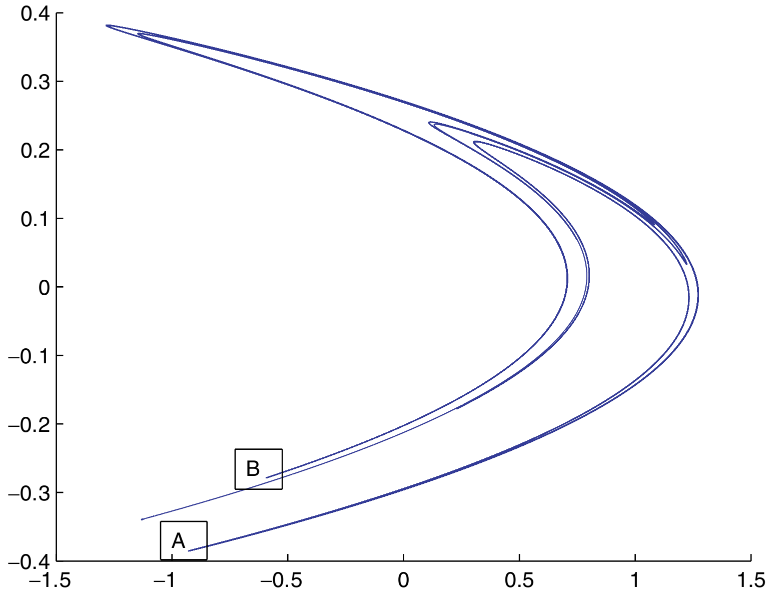
In order to determine the initial primary segment U_0 we approximate the unstable manifold near this saddle point as $y = p(x)$, where $p(x) = \sum_{i=0}^{10} c_i(x - x^*)^i$ with c_i computed from the relation (7). The initial primary segment U_0 is generated by $x_0 - x^* = 0.0001$, then $y_0 = p(x_0) = -0.339397140050439$.

[4]Note, this problem occurs in Carter's method, as well, but not in Hobson's.

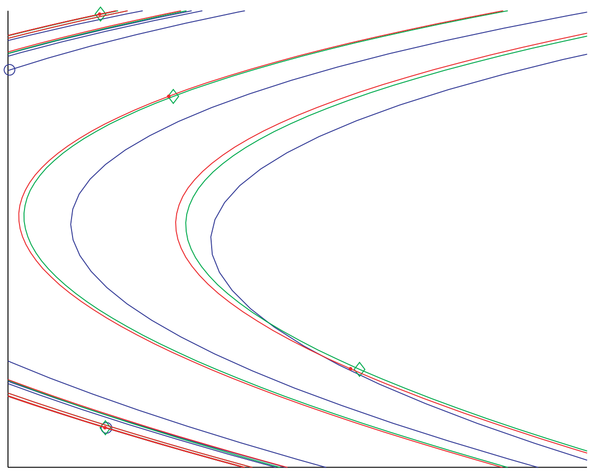
The last point of the initial primary segment is determined by $(x_0^1, y_0^1) = f(x_0, y_0) = (-1.131028508759507, -0.339376343126851)$. The error between this polynomial approximation and the true manifold along this segment is less than machine precision.

Figure 12(a) shows part of the computed unstable manifold of the Hénon Map: the union of primary segments U_0 through U_{20}. Figures 12(b) and 12(c) present a closeup view of the bounded boxes A and B from Fig. 12(a), respectively. They show more detail of three approximations of some sharp segments of the manifold, each approximation derived with a different tolerance condition. Additionally, the computed manifold is rotated in Figs. 12(b) and 12(c) and the aspect ratio is not preserved. These show the behavior of the method close to slanted hairpin turns.

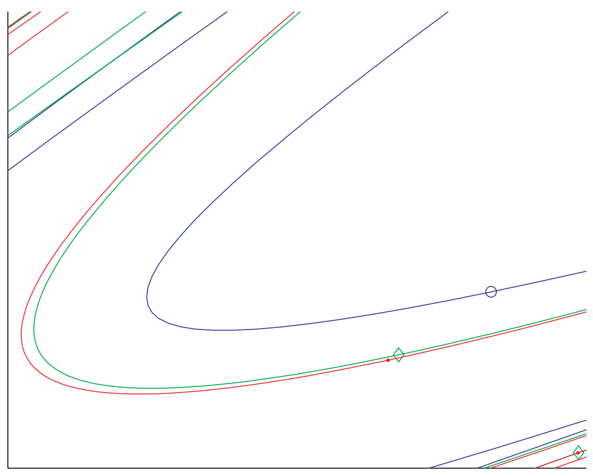
For computations of real unstable manifolds using methods based on linear interpolation, this concentration of points near maxima of curvature becomes a crippling fault. For example, the unstable



(a)



(b)



(c)

Fig. 12. (a) The unstable manifold of the Hénon Map, $b = 0.3$ and $a = 1.4$ with length about 419.6268, generated by the ACR3vs5 method; (b) closeup of box A with three approximations; (c) closeup of box B with three approximations, $\mathbf{tol} = 2^{-12}$–blue, $\mathbf{tol} = 2^{-14}$–green, $\mathbf{tol} = 2^{-16}$–red. Note subfigures (b) and (c) do not preserve the aspect ratio nor orientation of subfigure (a). In (b) $\mathbf{height} = O(10^{-5})$, $\mathbf{width} = O(10^{-4})$ and similarly for (c), $\mathbf{height} = O(10^{-6})$, $\mathbf{width} = O(10^{-4})$.

manifold of the Hénon Map develops folds where the curvature reaches $O(10^6)$. We found that ALI with such small tolerance places points so close together [e.g. see Fig. 10(b)] that, in 16-digit arithmetic, there are not enough significant digits remaining to meaningfully compute the angle between successive segments of the computed curve. See also Figs. 10 and 14(b).

Example 2. *McMillan Map*

Next, we consider the McMillan Map defined by

$$x' = y$$
$$y' = -x + 2y\left(\frac{\mu}{1+y^2} + \varepsilon\right).$$

For $\mu = 2.0$ and $\varepsilon = 0.05$ this map has a saddle fixed point at the origin. We approximate the unstable manifold near the origin as $p(x) = \sum_{i=0}^{10} c_i x^i$ with c_i computed using the relation (7).

In order to make direct comparison between the proposed methods and those of both Hobson and Carter, we test our example of the McMillan map, choosing the same parameter values and initial segment. The initial primary segment U_0 is generated by $x_0 = 0.001$, so that $y_0 = p(x_0) = 0.003839548998331$. The last point of the initial primary segment is determined by $(x_0^1, y_0^1) = f(x_0, y_0) = (0.003839548998331, 0.014741924483874)$. The error between this polynomial approximation and the true manifold along this segment is below machine precision.

Figure 13(a) shows part of the unstable manifold of the McMillan Map, computed using ACR3v5, the union of primary segments U_0 through U_{15}. Figures 13(b) and 13(c) present a closeup view of bounded boxes A and B from Fig. 13(a), respectively. They show more detail of three approximations of some sharp segments of the manifold, each approximation derived with different tolerance condition. These graphs show the method appears to converge.

8.1. *The proposed methods versus Hobson's and Carter's methods*

In order to compare the proposed methods with existing methods we also perform computations with the ALI+ method which is based on the same type of refinement condition as in both Hobson and Carter. Our implementation of ALI is slightly different from Carter's and we put some effort into making sure that ALI does not work any worse than in his studies; see Tables 1–3. We modify Carter's approach to the kink patch making sure that it always works.

In order to test convergence, Hobson considered the convergence of the arc length of an individual primary segment, estimating the length as the sum of the chord lengths between adjacent discrete points. We perform the same test and show the results in Tables 1–3, an expanded version of similar tables in [Carter, 2004] and [Hobson, 1993]. This test indicates that ACR3+ seems to be the fastest converging.

The sum of chord lengths between any neighboring points on the manifold does not converge to the length of the curve connecting them particularly fast. While the length of a cubic Bézier curve segment is not computable in closed form, a result due to [Gravesen, 1997] shows that estimating the arc length by one half the perimeter of the control polygon converges to the arc length faster than using chord length. Using this method, it is clear that the arc length of our methods converges even more rapidly. See Table 4.

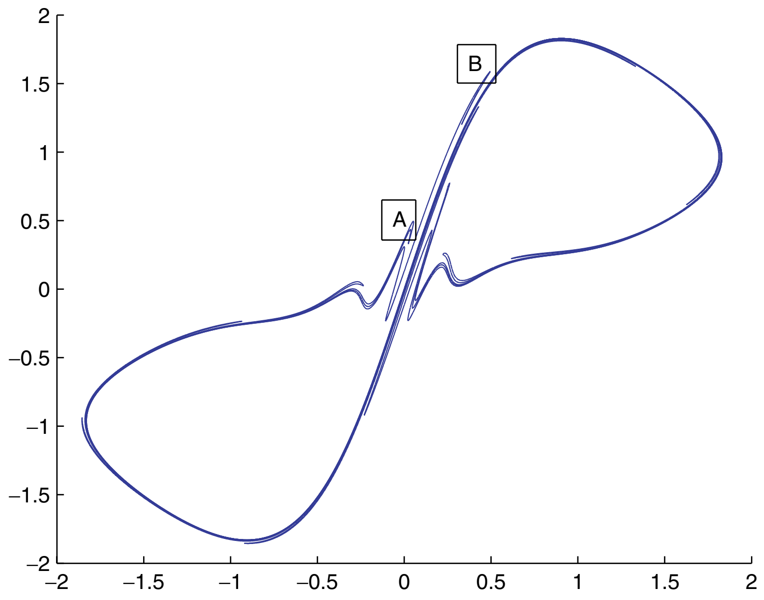
We do not believe that the arc length convergence is the best way to show the convergence of the method. Approximations to individual segments of the curve may approach the same length without lying close together.

8.2. *A more direct convergence test*

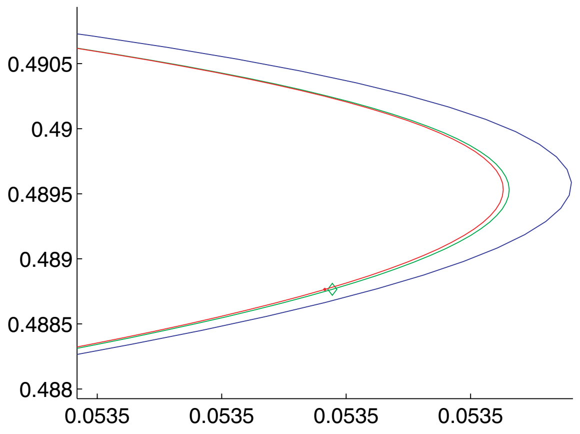
Figures 13(b) and 13(c) show that the ACR3vs5 method appears to converge. However, to make it more quantitative we perform further tests.

In order to show convergence of the proposed methods, we use high-order interpolation to generate continuous representations of the manifold for a very small value of the refinement condition; we call this an "exact" manifold. Next, we generate continuous representations of the manifold for several decreasing values of the refinement condition and check how the difference between the approximation and the "exact" manifold, measured using the maximum (L^∞) norm, decays. Note that this is the same test as we performed in Sec. 6 for the model problem; see Fig. 10.

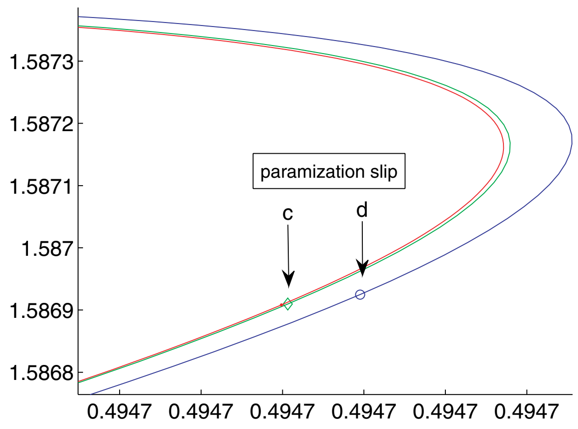
The initial primary segment U_0 is generated as above and the first 15 primary segments are computed. Figure 14(a) shows the convergence of the two proposed methods, ACR3vs5 and ACR3+. Additionally, as a comparison, the figure presents convergence of ALI+, described in the previous



(a)



(b)



(c)

Fig. 13. (a) The unstable manifold of the McMillan Map, $\mu = 2.0$ and $\varepsilon = 0.05$ with length about 113.3335, generated by ACR3vs5; (b) close up of box A; (c) close up of box B with three approximations, $\textbf{tol} = 2^{-11}$ — blue, $\textbf{tol} = 2^{-13}$ — green, $\textbf{tol} = 2^{-15}$ — red. Note (b) and (c) do not preserve the aspect ratio of (a), for (b) $\textbf{height} = O(10^{-3})$, $\textbf{width} = O(10^{-5})$ and for (c) $\textbf{height} = O(10^{-4})$, $\textbf{width} = O(10^{-4})$.

Table 1. Manifold calculation comparison using chord arc length of primary segments of the McMillan Map. The bold type values we can call the "exact" lengths of the segments. Values for Hobson's and Carter's method follow [Carter, 2004].

	Hobson's Method				Carter's Method				
tol$_1$	0.45	0.45	0.45	0.15	0.1	0.07	0.05	0.03	0.01
tol$_2$	0.01	0.003	0.001	0.00001					
U_{13}	20.020	20.117	20.119	20.120	20.115	20.115	20.117	20.118	20.120
U_{14}	22.834	23.098	23.097	23.098	22.975	23.090	23.092	23.094	23.097
U_{15}	28.407	28.991	29.034	29.037	28.687	29.024	29.033	29.037	29.038
Calls	4975	7671	11 395	90 813	3144	4364	5837	9578	27 493

Table 2. Arc length values for ALI+ method computed using ten points at the initial segment.

	ALI+							
tol	0.143	0.102	0.0737	0.0555	0.0445	0.0365	0.0145	0.0044
U_{13}	20.1141	20.1169	20.1183	20.1190	20.1193	20.1195	20.1199	**20.1200**
U_{14}	23.0882	23.0938	23.0956	23.0965	23.0969	23.0972	23.0976	**23.0977**
U_{15}	29.0154	29.0304	29.0350	29.0366	29.0371	29.0375	29.0379	**29.0381**
Calls	3126	4351	5832	7664	9579	11 407	27 503	90 703

Table 3. Arc length values for ACR3+ and ACR3vs5 computed using ten points at the initial segment.

	ACR3+				ACR3vs5			
tol	$2^{(-9)}$	$2^{(-12)}$	$2^{(-14)}$	$2^{(-16)}$	$2^{(-13)}$	$2^{(-18)}$	$2^{(-20)}$	$2^{(-23)}$
U_{13}	20.1136	20.1192	20.1198	20.1199	20.1160	20.1195	20.1198	20.1199
U_{14}	23.0903	23.0963	23.0974	23.0976	23.0920	23.0970	23.0974	23.0976
U_{15}	29.0210	29.0357	29.0375	29.0380	29.0234	29.0367	29.0375	29.0380
Calls	1666	4276	8397	16 649	2223	7035	10 559	20 988

Table 4. Manifold calculation comparison using Gravesen's arc length approximation of primary segments of McMillan Map. The values are computed for the given value of the tolerance with ten points at the initial primary segment.

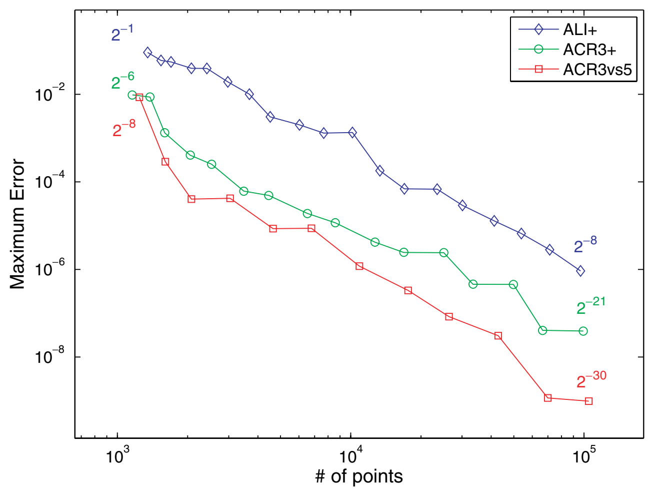
	ACR3+				ACR3vs5			
tol	$2^{(-8)}$	$2^{(-9)}$	$2^{(-10)}$	$2^{(-13)}$	$2^{(-7)}$	$2^{(-10)}$	$2^{(-13)}$	$2^{(-17)}$
U_{13}	20.1190	20.1195	20.1199	20.1200	20.1199	20.1203	20.1199	20.1200
U_{14}	23.0931	23.0975	23.0977	23.0977	23.0942	23.1003	23.0983	23.0977
U_{15}	29.0286	29.0359	29.0383	29.0381	29.0167	29.0400	29.0375	29.0381
Calls	1190	1666	2219	6246	666	1166	2223	5314

paragraph, the method based on the angle refinement condition.

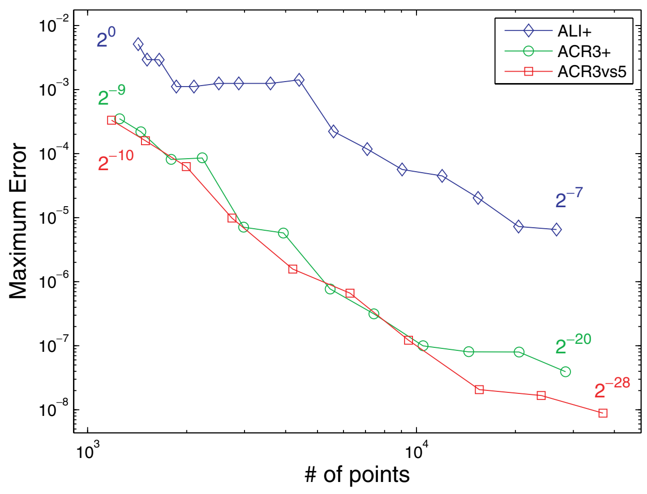
We perform the same test for the Hénon map. We use the same initial primary segment U_0 as above, and compute the first 16 primary segments, [Fig. 12(a) contains 20 segments]. Figure 14(b) shows that both methods converge very well and at a similar rate. The error between approximation and the "exact" manifold with about 10^5 points is on the order of 10^{-8}. The ALI method fails to converge for **tol** $< 2^{-8}$ with 16 iterates of the map. Using this tolerance, the algorithm also failed to converge on the 17th iterate, as discussed in the final paragraph in the discussion of Example 1.

Figures 14(a) and 14(b) show that both proposed methods (ACR3vs5 and ACR3+) converge faster than ALI+. The method ACR3vs5 seems to work slightly better than the ACR3+. However, for a few values of tolerance the second method outperforms the first.

We encountered a difficulty in measuring the error in the unstable manifold calculations not present in the parametric curve convergence studies in Sec. 6. Since we do not know the true manifold W^{u}, we compute the error by calculating the distance between two approximations of W^{u}. Because the map is expanding in the direction tangent to W^{u}, calculations with different values of the tolerance lead to slight changes in the parameterization of the curve. Plotting, say, the x-coordinate as a function of the parameter t, we find that the two graphs look the same but are shifted slightly along the t-axis. We call this phenomenon *parameterization slip* and, although it is the largest source of our computed error, it does not correspond to an actual error in the computation of the manifold. It is illustrated in Fig. 13(c): the points marked **c** and **d** correspond to the same parameter value, obtained using different values of the tolerance. The calculated error represents the distance



(a)



(b)

Fig. 14. Convergence of the adaptive methods ACR3vs5 (blue), ACR3+ (red) and ALI+ (green) with decreasing refinement conditions changing as indicated, between the values at the opposite ends of each graph; 50 points on the initial primary segment. (a) Test on the McMillan Map. The "exact" manifolds for ALI+, ACR3+ and ACR3vs5 with tolerances 2^{-9}, 2^{-22} and 2^{-32}, respectively. (b) Test on the Hénon Map. The "exact" manifolds for ALI+ ACR3+ and ACR3vs5 with tolerances 2^{-8}, 2^{-21} and 2^{-30}, respectively, without reparameterization.

between these two points, which is clearly much larger than the distance between the two curves in this neighborhood.

The results in Figs. 14(a) and 14(b) are slightly different than the results which we see in Fig. 10(a) for the model problem. However, computation of these manifolds is more complex than the simple problem of drawing a parametric curve. The accuracy of that computation depends only on the interpolation error, whereas the adaptive method for computing the invariant manifold has several sources of error. The convergence of the method can be affected by the interpolation error and by the amount by which this error and round-off errors are amplified by sensitive dependence on initial conditions. Much of the observed error is actually due to parameterization slip. The error in the normal direction often appears to be significantly smaller. In the last case, we can partially solve the problem of parameterization slip by chord length reparameterization. However, our numerical tests show that this does not always help. The reparameterization improved the convergence for the McMillan map, whereas for the Hénon map it did not.

Example 3. *A map with an explicit manifold*

In this section, we test the methods on a map constructed explicitly to have an unstable manifold that is computable in closed form. $W^{\mathrm{u}} = \{(x, y) \mid y = h(x)\}$. First, since y is an explicit function of x, we avoid the "parameterization slip" and its effect on the error. Second, we are able to compute the exact error between the true and computed manifolds, rather than the distance between two approximations. Note that the errors reported in this section refer to distance in the y direction rather than Euclidean distance. A similar test on an explicit manifold is performed in [Osinga & Rokni, 2005].

We consider the example for which the unstable manifold can be found explicitly, namely

$$\begin{aligned}\dot{x} &= x\\ \dot{y} &= -y + g(x),\end{aligned} \tag{8}$$

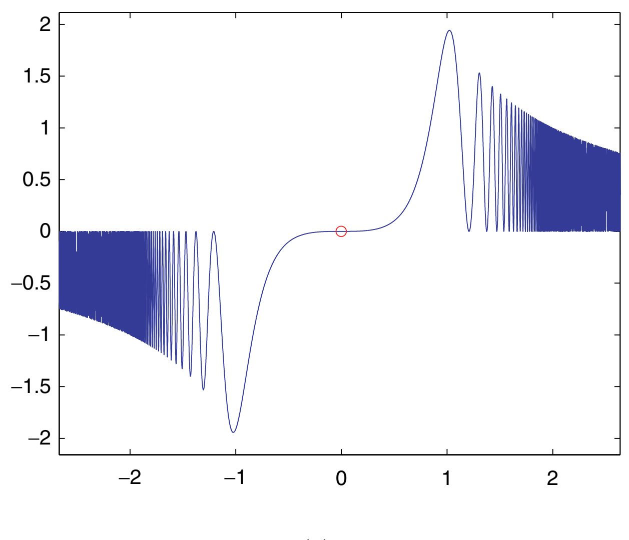
with explicit solution

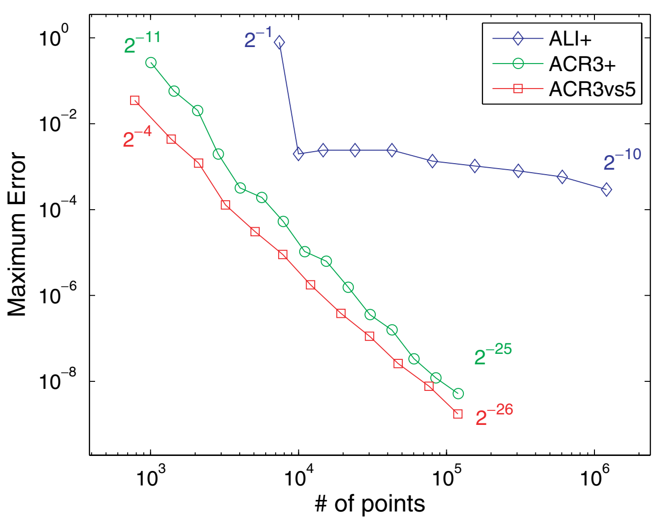
$$\begin{aligned}x(t) &= c_1 e^t\\ y(t) &= c_2 e^{-t} + \frac{G(c_1 e^t)}{c_1 e^t},\end{aligned} \tag{9}$$

where the function $G(x)$ satisfies $G'(x) = g(x)$. We assume that $g(0) = 0$, i.e. $G'(0) = 0$, this automatically ensures that the origin is a saddle point with associated stable and unstable manifolds tangent to the x and y axes, respectively. Note from (9) that trajectories of the system can be explicitly written in the form

$$y = \frac{c_1 c_2 + G(x)}{x}.$$

Note that assuming the explicit unstable manifold is tangent to the y axis at the origin


(a)



(b)

Fig. 15. (a) The explicit unstable manifold of the system (8) given by Eq. (10), (b) convergence of the error between the manifold and approximations given by adaptive methods ACR3vs5 (red), ACR3+ (green) and ALI+ (blue), the test performed on the branch with length about 255, the initial primary segment starts at $x = 0.0011$ with 10 points and the first 11 primary segments are computed. Decreasing refinement conditions as indicated, between the values at the opposite ends of each graph. (a) The unstable manifold. (b) Convergence of the three methods.

requires $y(0) = y'(0) = 0$. Assuming $c_1c_2 = 1$, the function $G(x) = -\cos(x^2e^{x^2})$ satisfies all the conditions above and guarantees the existence of the explicit unstable manifold of the form

$$y = \frac{1 - \cos(x^2e^{x^2})}{x}; \tag{10}$$

see Fig. 15(a). In addition, it is significantly complex to providing a challenging test for the various algorithms.

In order to test the proposed methods we need to find the planar map associated with the system (8). For any flow $\varphi^t(x, y)$ given by the solution of a dynamical system, the planar map can be written as

$$(x', y') = f(x, y) = \varphi^T(x, y),$$

where T is a constant. Consider the flow $\varphi^t(x, y)$ given by (9) with $T = \ln 2$, then the planar map takes the form

$$\begin{aligned} x' &= 2x \\ y' &= \frac{xy + G(2x) - G(x)}{2x}. \end{aligned} \tag{11}$$

where the function $G(x)$ is as defined above.

We apply each of the proposed methods to the map (11) to generate the rightgoing branch of the unstable manifold; see Fig. 15(a). Next we investigate the behavior of the maximum error between the exact manifold and each approximation for decreasing refinement condition.

Figure 15(b) shows the convergence of the error for each method. Both methods, ACR3+ and ACR3vs5, perform very well (here, it appears ACR3vs5 converges slightly faster), whereas the ALI+ method does poorly. The test above confirms our previous results for the proposed methods. The question which method, ACR3+ or ACR3vs5, for computing unstable manifold is better in general remains open.

9. Discussion

The methods presented here for computing unstable manifolds incorporate ideas from computer aided geometric design and achieve significant improvements in the accuracy of the calculation, and in reducing the number of calls to the map f. This is especially relevant if one wants to resolve all the sharp folds typically found on such manifolds. For the mere display of the manifold on a screen, resolving points where the curvature reaches such maxima would be excessive, but for computations involving the manifolds, such as estimating their dimension as suggested by [Hobson, 1993], or calculating their intersections in order to apply the theory of homotopic lobe dynamics [Mitchell & Delos, 2006], this would be necessary.

We should point out, however, that there are other costs involved with the implementation of this method, in comparison, say, with the simpler method in [Carter, 2004]. In particular, one segment of a piecewise linear interpolant can be plotted using only its values at the two endpoints, with high-level graphing software filling in the points in between. To effectively render a cubic interpolant, one first samples the curve at a finite number of points in between those interpolated, and then plots a linear interpolant through those points. By choosing these points adaptively, one can plot the piecewise-cubic curve to the desired degree of accuracy, but the overhead of this adaptive calculation makes the procedure more expensive than simply evaluating the interpolant of a finite number of equally spaced parameter values. Here, too, one has the choice to either choose a large enough number of points to guarantee this will accurately compute the curve, or else to estimate the necessary number of points using a bound based on the derivatives of the interpolant; see, for example, references [Cheng, 1992; Filip *et al.*, 1986].

The method has some additional advantages over methods based on linear interpolation. To analyze transport in a chaotic system, one needs to construct regions bounded by segments of the stable and unstable manifolds, and thus must determine intersections of $W^{\rm s}$ and $W^{\rm u}$. There are two steps to this process. Given piecewise interpolants to the two curves, one first compiles a list of pairs of segments whose bounding boxes overlap and which might intersect. Second, one tests each of these pairs for an intersection. In the second step, it is simpler to detect intersections between linear interpolants (solving a small system of linear equations), although many efficient methods exist for calculating the intersections of Bézier curves, for example [Sederberg, 1989; Sederberg & Nishita, 1990]. It is the first step that takes the most time however. Given n segments, there are $n(n-1)/2$ pairs of segments to check. Since our method greatly reduces n, finding the candidate pairs will take significantly less time.

While discussing manifold intersections, we should also point out that, as the map f is invertible, $W^{\rm u}$ is topologically forbidden from self-intersecting. A more complete code should also check for self-intersections and refine the manifolds in their neighborhoods. Due to the fractal nature of the strange attractor, arbitrarily many additional segments of the manifold pass within any small neighborhood of any point on $W^{\rm u}$. Because of the finite errors in the method, and the arbitrarily small spacing between segments, a computed manifold based on any refinement conditions of the type discussed will possess erroneous self-intersections.

We would also like to point out some complementary work. Mireles-James [2009] adapted as a computational tool the parameterization method of Cabré *et al.*, developed as a rigorous tool in the study of invariant manifolds [Cabré *et al.*, 2003a, 2003b, 2005], and previously applied in [Franceschini & Russo, 1981] as part of a numerical pronumber of chaos for the Hénon map. We introduced this method briefly in Sec. 3. This method avoids the roundoff error and interpolation error we find in our methods by representing the solution as a power series in a parameter s, and eliminates the need to iterate fundamental segments. On the other hand, this method generates a parametric form of $W^{\rm u}$. The methods described in Sec. 5.2 are ideally suited to sample this curve efficiently.

In future work, we plan to extend these methods to the computation of two-dimensional invariant manifolds in three-dimensional maps. This problem is harder for a variety of reasons, well-summarized in [Krauskopf & Osinga, 1998a]. Most importantly, the hyperbolic fixed point can have eigenvalues $\lambda_1 > \lambda_2 > 1 > \lambda_3$. Iterating a fundamental segment (in this case an annular region) will lead to greater growth in the λ_1 direction, and the computed manifold will be severely stretched in this manner, having an aspect ratio on the order of $(\lambda_1/\lambda_2)^n$ after n iterations of the map. To avoid this problem, their method works by growing the manifold radially from the fixed point in such a way that the growth rate is nearly constant in all directions (their method in [Krauskopf & Osinga, 1998b] for one-dimensional maps works much the same way). To achieve this, they must, as Hobson does, discard certain calculated points until a point of proper distance from the edge is found. This method seems fundamentally at odds with our approach of calculating the image of an entire segment and of using high-order error conditions to build a bisection method.

Finally, we would like to make the point that we have not seen many applications of CAGD methods to dynamics problems in this manner before, and that we believe there is wide potential for their adoption. One exception is the work by Henderson on numerical methods of invariant manifolds of continuous-time problems, although he does use a different set of tools than those described here [Henderson, 2005, 2006, 2009]. CAGD methods, especially NURBS, have also been widely used in the finite elements literature. A large and very accessible literature exists on this subject, and the methods we have used here are not particularly sophisticated. We hope that these techniques can prove useful to other researchers in dynamical systems. Two excellent books are [Farin, 2002] and [Goldman, 2003].

Acknowledgments

The authors would like to thank Denis Blackmore, Josh Carter, Rafael de la Llave, Hector Lomelí, Jason Mireles-James, Hinke Osinga, and Denis Zorin for helpful discussions. NJIT undergraduates Matthew Peragine and Fatima Elgammal collaborated on a preliminary version of this work as part of the CSUMS program, supported by NSF DMS-0639270. The authors both received support from NSF DMS-0807284.

References

Cabré, X., Fontich, E. & de la Llave, R. [2003a] "The parameterization method for invariant manifolds I: Manifolds associated to non-resonant subspaces," *Indiana U. Math. J.* **52**, 283–328.

Cabré, X., Fontich, E. & de la Llave, R. [2003b] "The parameterization method for invariant manifolds II: Regularity with respect to parameters," *Indiana U. Math. J.* **52**, 329–360.

Cabré, X., Fontich, E. & de la Llave, R. [2005] "The parameterization method for invariant manifolds III: Overview and applications," *J. Diff. Eqs.* **218**, 444–515.

Carter, J. [2004] "A bisection method for computing invariant manifolds of 2-D maps," Preprint, posted, with the author's permission, at http://web.njit.edu/˜goodman/publications/carter.pdf.

Chandler, R. [1990] "A recursive technique for rendering parametric curves," *Comput. Graph.* **14**, 477–479.

Cheng, F. [1992] "Estimating subdivision depths for rational curves and surfaces," *ACM T. Graph.* **11**, 140–151.

Collins, P. [2002] "Symbolic dynamics from homoclinic tangles," *Int. J. Bifurcation and Chaos* **12**, 605–617.

de Figueiredo, L. H. [1995] "Adaptive sampling of parametric curves," *Graphics Gems V*, 173–178.

England, J., Krauskopf, B. & Osinga, H. [2005] "Computing one-dimensional global manifolds of Poincaré maps by continuation," *SIAM J. Appl. Dyn. Syst.* **4**, 1008–1041.

Farin, G. [2002] *Curves and Surfaces for CAGD: A Practical Guide*, 5th edition (Morgan-Kaufmann, San Francisco, CA).

Filip, D., Magedson, R. & Markot, R. [1986] "Surface algorithms using bounds on derivatives," *Comput. Aided Geom. D.* **3**, 295–311.

Franceschini, V. & Russo, L. [1981] "Stable and unstable manifolds of the Henón mapping," *J. Stat. Phys.* **25**, 757–769.

Gamet, L., Ducros, F., Nicoud, F. & Poinsot, T. [1999] "Compact finite diffrence scheme for non-uniform mesh. Aplication to direct numerical symulation of compressible flows," *Int. J. Numer. Meth. Fluids* **29**, 159–191.

Goldman, R. [2003] *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling* (Morgan-Kaufmann).

Goodman, R., Holmes, P. & Weinstein, M. [2002] "Interaction of sine-Gordon kinks with defects: Phase space transport in a two-mode model," *Phys. D* **161**, 21–44.

Goodman, R., Holmes, P. & Weinstein, M. [2004] "Strong NLS soliton-defect interactions," *Phys. D* **192**, 215–248.

Goodman, R. [2008] "Chaotic scattering in solitary wave interactions: A singular iterated-map description," *Chaos* **18**, 023113.

Gravesen, J. [1997] "Adaptive subdivision and the length and energy of Bézier curves," *Comp. Geom.-Theor. Appl.* **8**, 13–31.

Henderson, M. E. [2005] "Computing invariant manifolds by integrating fat trajectories," *SIAM J. Appl. Dyn. Syst.* **4**, 832–882.

Henderson, M. E. [2006] "Covering an invariant manifold with fat trajectories," *Model Reduction and Coarse-Graining Approaches for Multiscale Phenomena* (Springer, Berlin), pp. 39–54.

Henderson, M. E. [2009] "Flow box tiling methods for compact invariant manifolds," preprint, pp. 1–25.

Hénon, M. [1976] "A two-dimensional mapping with a strange attractor," *Comm. Math. Phys.* **50**, 69–77.

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

Kostelich, E., Yorke, J. & You, Z. [1996] "Plotting stable manifolds: Error estimates and noninvertible maps," *Phys. D* **93**, 210–222.

Krauskopf, B. & Osinga, H. [1998a] "Globalizing two-dimensional unstable manifolds of maps," *Int. J. Bifurcation and Chaos* **8**, 483–503.

Krauskopf, B. & Osinga, H. [1998b] "Growing 1D and quasi-2D unstable manifolds of maps," *J. Comput. Phys.* **146**, 404–419.

Krauskopf, B. & Osinga, H. [1999] "Two-dimensional global manifolds of vector fields," *Chaos* **9**, 768–774.

Krauskopf, B., Osinga, H., Doedel, E., Henderson, M., Guckenheimer, J., Vladimirsky, A., Dellnitz, M. & Junge, O. [2004] "A survey of methods for computing (un)stable manifolds of vector fields," *Int. J. Bifurcation and Chaos* **15**, 763–792.

Meiss, J. [1997] "Average exit time for volume-preserving maps," *Chaos* **7**, 139–147.

Mireles-James, J. [2009] "Elementary example of the parametrization method. stable and unstable manifolds of the standard map," preprint, University of Texas at Austin.

Mireles-James, J. & Lomelí, H. [2010] "Computation of heteroclinic arcs with application to the volume preserving Hénon family," *SIAM J. Appl. Dyn. Syst.* (*accepted*) .

Mitchell, K. & Delos, J. [2006] "A new topological technique for characterizing homoclinic tangles," *Phys. D* **221**, 170–187.

Mitchell, K. [2009] "The topology of nested homoclinic and heteroclinic tangles," *Phys. D* **238**, 737–763.

Osinga, H. & Rokni, R. [2005] "Numerical study of manifold computations," *Equadiff 2005*, eds. Dumortier, F., Broer, H., Mawhin, J., Vanderbauwhede, A. & Lunel, S. V. (World Scientific, Singapore), pp. 190–195.

Parker, T. & Chua, L. [1989] *Practical Numerical Algorithms for Chaotic Systems* (Springer).

Rom-Kedar, V. [1990] "Transport rates of a class of two-dimensional maps and flows," *Phys. D* **43**, 229–268.

Rom-Kedar, V. [1994] "Homoclinic tangles–classification and applications," *Nonlinearity* **7**, 441–473.

Sederberg, T. [1989] "Algorithm for algebraic curve intersection," *Comput. Aided Design* **21**, 547–554.

Sederberg, T. & Nishita, T. [1990] "Curve intersection using Bézier clipping," *Comput. Aided Design* **22**, 538–549.

Tzafestas, S. & Pantazopoulos, J. [1999] "An efficient algorithm for rendering parametric curves," *Advances in Intelligent Systems*: *Concepts*, *Tools*, *and Applications*, ed. Tzafestas, S. (Springer), pp. 357–366.

You, Z., Kostelich, E. & Yorke, J. [1991] "Calculating stable and unstable manofolds," *Int. J. Bifurcation and Chaos* **1**, 605–623.

Appendix A

Detailed Introduction to CAGD Tools

Here we introduce a few tools from CAGD that are used to construct the numerical method.

A.1. *Piecewise linear interpolation*

For a given sequence of points $\mathbf{x}_0, \ldots, \mathbf{x}_n$ and parameter values $t_0, \ldots, t_n$, the line segment between the points $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ can be written as

$$s_k(t) = \frac{t_k - t}{t_k - t_{k-1}}\mathbf{x}_{k-1} + \frac{t - t_{k-1}}{t_k - t_{k-1}}\mathbf{x}_k \quad \text{for } t \in [t_{k-1}, t_k]. \tag{A.1}$$

Therefore, the whole interpolated curve can be written as

$$\gamma_{\text{approx}} = \bigcup_{k=1}^{n} s_k(t).$$

To accurately approximate a smooth curve using linear interpolation requires a large number of points, especially near regions of large curvature.

We can estimate the error of the linear interpolation. For any C^2 curve such that $\gamma : [t_{k-1}, t_k] \to \mathbb{R}^n$ and the linear interpolant given by Eq. (A.1),

$$\sup_{t\in[t_{k-1},t_k]} \|\gamma(t) - s_k(t)\| \leq \frac{(t_k - t_{k-1})^2}{8} \sup_{t\in[t_{k-1},t_k]} \|\gamma''(t)\|;$$

see the proof in [Filip *et al.*, 1986].

A.2. *Bézier curves*

The fundamental object in geometric modeling is the Bézier curve, which can be constructed using the Bernstein polynomials. The Bernstein polynomials are given by

$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}.$$

Note that, by the binomial theorem,

$$1 = (t + (1-t))^n = \sum_{k=0}^{n} B_k^n(t). \tag{A.2}$$

A Bézier curve has parametric form given by the convex affine combination (i.e. with positive weights summing to one) of $n+1$ *control points* $\mathbf{p}_0, \ldots, \mathbf{p}_n$:

$$\beta(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{p}_k. \tag{A.3}$$

Note that $\beta(0) = \mathbf{p}_0$ and $\beta(1) = \mathbf{p}_n$, but that the other control points are not interpolated by the curve. We can see from Eq. (A.2) that for $0 \leq t \leq 1$, the point $\beta(t)$ is a weighted average of the control points, and as the weights defined by the Bernstein polynomials are positive, this curve must lie inside the convex hull of the control points. The polygon $\mathbf{P}$ formed by $\mathbf{p}_0, \ldots, \mathbf{p}_n$ is called the *Bézier polygon* or *control polygon* of the curve. Note that in general the edge of a convex hull (the bounding box) is not the same as the control polygon. An example of a cubic Bézier curve is shown in Fig. 16.

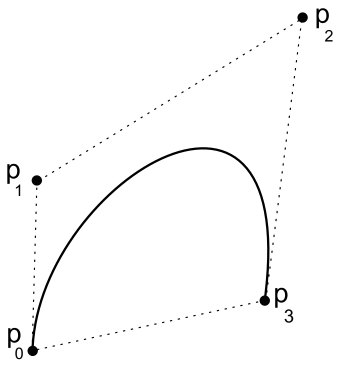


Fig. 16. A cubic Bézier curve, together with its control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$, and their convex hull.

Bézier curves are invariant under affine transformation of the independent variable t; setting

$$t = \frac{u-a}{b-a}, \tag{A.4}$$

then the curve $\gamma(t(u))$ over the interval $[a, b]$ parameterizes the same curve as $\gamma(t)$ over $[0, 1]$.

Finally, note from Eq. (A.3) that the tangent vectors to the curve at the endpoints $\mathbf{p}_0$ and $\mathbf{p}_n$ are given by

$$\mathbf{T}_0 = n(\mathbf{p}_1 - \mathbf{p}_0) \quad \text{and} \quad \mathbf{T}_n = n(\mathbf{p}_n - \mathbf{p}_{n-1}), \tag{A.5}$$

respectively — this formula is modified slightly if the parameterization in Eq. (A.4) is used. This can be observed in Fig. 16.

Bézier curves are widely used in CAGD because, in addition to the above properties, there exist efficient algorithms for evaluating them (forward-differencing and the de Casteljau algorithm), and for performing other calculations such as finding their intersections [Sederberg, 1989; Sederberg & Nishita, 1990] or their arc length [Gravesen, 1997].

A.3. *Composite Bézier curves*

Several Bézier curves may be pieced together in order to generate shapes that are too complex for a single low-degree Bézier curve to handle. (High degree Bézier curves suffer from Runge's Phenomenon.) In concatenating Bézier curves, we need to control the smoothness of the composite curve. Let $\mathbf{p}_0, \ldots, \mathbf{p}_3$ and $\mathbf{p}_3, \ldots, \mathbf{p}_6$ be the Bézier points of two cubic curve segments $\mathbf{P}_{[a,c]}$ and $\mathbf{Q}_{[c,b]}$; see Fig. 17. Since they share the point $\mathbf{p}_3$, their union clearly forms a continuous, or C^0, curve. With this minimal continuity requirement, the two curves may form a corner. To ensure that the two pieces meet smoothly, more care is called for.

Two adjacent curve segments $\mathbf{P}$ and $\mathbf{Q}$ are said to be C^k continuous (or, to have kth order parametric continuity) if

$$\mathbf{P}(c) = \mathbf{Q}(c), \mathbf{P}'(c) = \mathbf{Q}'(c), \ldots, \mathbf{P}^{(k)}(c) = \mathbf{Q}^{(k)}(c).$$

Thus, C^0 means simply that the two adjacent curves share a common endpoint, $\mathbf{p}_3$ in our case. C^1 means that the two curves not only share the same endpoint, but also that they have the same first order parametric derivatives. C^2 means that two curves are C^1 and in addition that they have the same second order parametric derivatives at their shared endpoint. Equation (A.5) demonstrates that this curve is C^1 if and only if $(\mathbf{p}_3 - \mathbf{p}_2)/(c-a) = (\mathbf{p}_4 - \mathbf{p}_3)/(b-c)$. Similar condition exists for C^2 and higher continuity.

A slightly weaker notion of continuity of a piecewise curve, one which is independent of parameterization, is called geometric continuity and is denoted G^k. The two curves are G^k continuous at $\mathbf{p}_3$ if the kth derivative vectors from both sides point in the same direction. In practice, this means that the two component curves may be reparameterized to make their union C^k. The conditions for geometric continuity (also known as visual continuity) are less strict than for parametric continuity.

G^1 continuity requires that the three points $\mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$ are collinear; i.e. they have a common tangent line at their shared endpoint. G^2 (second order visual or geometric continuity) means that the two neighboring curves have the same tangent line and also the same curvature at their common

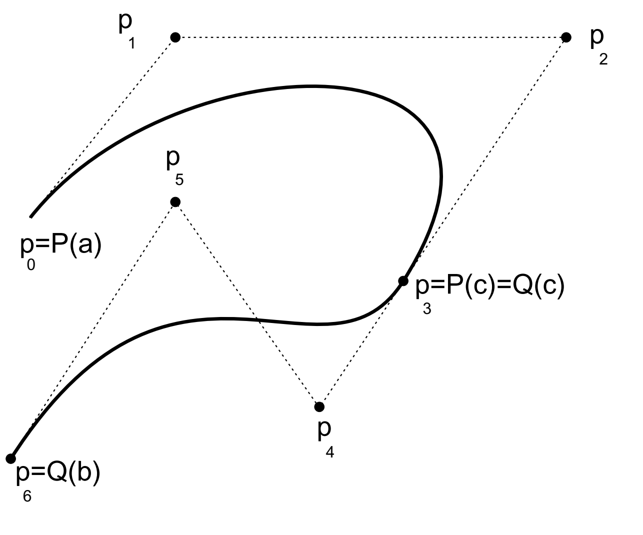


Fig. 17. Composite of two Bézier curves.

boundary. Clearly, two curves which are C^n are also G^n.

A.4. *The de Casteljau algorithm*

This is a recursive method to evaluate a Bézier curve $\mathbf{P}_{[a,b]}$ at an arbitrary parametric location $t \in (a,b)$. As an example, the geometric interpretation of the evaluation algorithm for a point on cubic Bézier curve is shown in Fig. 18. We label the control points of a cubic Bézier curve $\mathbf{P}_{[a,b]}$ with $\mathbf{p}_0^0$, $\mathbf{p}_1^0$, $\mathbf{p}_2^0$, and $\mathbf{p}_3^0$. Each line segment in the trellis is split in the ratio $(t-a)/(b-t)$, i.e. we define

$$\mathbf{p}_j^{i+1} = \frac{((t-a)\mathbf{p}_j^i + (b-t)\mathbf{p}_{j+1}^i)}{(b-a)}. \tag{A.6}$$

The value $\mathbf{P}_{[a,b]}(t) = \mathbf{p}_0^3$ is the value of $\beta(t)$ at t as defined in Eq. (A.3).

The de Casteljau algorithm can also be used to subdivide a Bézier curve $\mathbf{P}_{[a,b]}$ into two shorter Bézier curves $\mathbf{P}_{[a,t]}$ and $\mathbf{P}_{[t,b]}$ whose union is equivalent to $\mathbf{P}_{[a,b]}$. The control points for $\mathbf{P}_{[a,t]}$ are $\mathbf{p}_0^0$, $\mathbf{p}_0^1$, $\mathbf{p}_0^2$, $\mathbf{p}_0^3$ and the control points for $\mathbf{P}_{[t,b]}$ are $\mathbf{p}_0^3$, $\mathbf{p}_1^2$, $\mathbf{p}_2^1$, $\mathbf{p}_3^0$. This is not the most efficient computational algorithm to compute $\mathbf{P}_{[a,b]}$ but is very important in the mathematical theory of such curves.

A.5. *Hermite interpolating polynomials*

Bézier curve, by themselves, are insufficient to construct an interpolation scheme. Given two points,

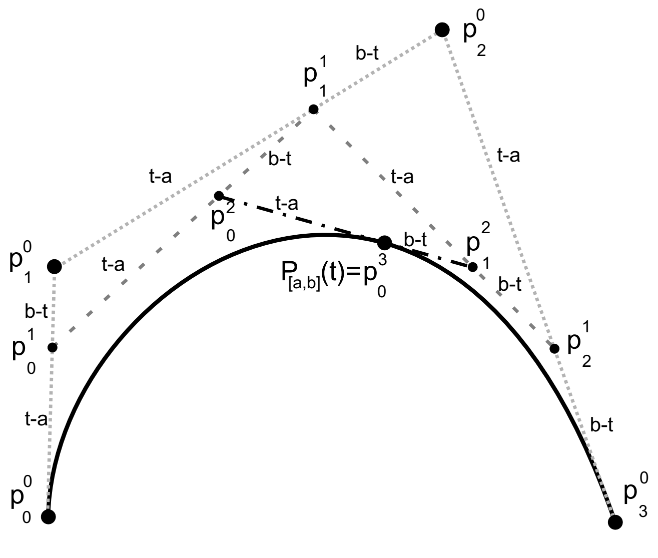


Fig. 18. The De Casteljau construction for evaluating/subdividing a cubic Bézier curve.

$\mathbf{x}_1$ and $\mathbf{x}_2$ and two vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, the *cubic Hermite interpolating polynomial* is the unique cubic polynomial $\mathbf{p}(t)$ that interpolates the two points at $t=0$ and $t=1$, respectively, with tangent vectors $\mathbf{v}_1$ at $\mathbf{x}_1$ and $\mathbf{v}_2$ at $\mathbf{x}_2$. Using relation (A.5), the Hermite interpolating polynomial can be written as a cubic Bézier curve with

$$\begin{gathered}\mathbf{p}_0 = \mathbf{x}_1, \quad \mathbf{p}_1 = \mathbf{x}_1 + \frac{\mathbf{v}_1}{3}, \\ \mathbf{p}_2 = \mathbf{x}_2 - \frac{\mathbf{v}_2}{3}, \quad \text{and} \quad \mathbf{p}_3 = \mathbf{x}_2.\end{gathered} \tag{A.7}$$

It should be noted that, unlike Bézier curves, Hermite interpolating polynomials are not invariant under affine changes of the parameter t, given by Eq. (A.4). Such a change of variables changes the length of the tangent vectors and thus produces a different curve. To preserve the shape of the curve, the tangent vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ must be scaled appropriately.

If a curve $\gamma(t) = (x(t), y(t))$ is approximated by its Hermite interpolant, with $\mathbf{v}_1$ and $\mathbf{v}_2$ given by the exact tangent vectors at the endpoints, the error for $t \in [t_{k-1}, t_k]$ is given by

$$\begin{aligned} x(t) - p_1(t) & \\ &= \frac{x^{(4)}(c)}{4!}(t-t_{k-1})^2(t-t_k)^2 \\ &\qquad \text{for some } c \in [t_{k-1}, t_k], \end{aligned}$$

the maximum of the right-hand side is achieved for $t = (t_{k-1}+t_k)/2$, so we can bound the error in the first component for $t \in [t_{k-1}, t_k]$ with the expression

$$\begin{aligned} &\sup_{t\in[t_{k-1},t_k]} |x(t) - p_1(t)\| \\ &\qquad \le \frac{(t_k - t_{k-1})^4}{384} \sup_{t\in[t_{k-1},t_k]} |x^{(4)}(t)\|. \end{aligned}$$

The error in the y component has a similar bound. Then, we can estimate the error of the Hermite interpolant by

$$\begin{aligned} &\sup_{t\in[t_{k-1},t_k]} \|\gamma(t) - \mathbf{p}(t)\| \\ &\qquad \le \frac{(t_k - t_{k-1})^4}{384} \left(\left(\sup_{t\in[t_{k-1},t_k]} |x^{(4)}(t)| \right)^2 \right. \\ &\qquad\qquad \left. + \left(\sup_{t\in[t_{k-1},t_k]} |y^{(4)}(t)| \right)^2 \right)^{1/2}. \end{aligned}$$

This guarantees that refining the set of parameter values, results in better approximation of the parametric curve.

A.6. *Catmull–Rom splines*

Catmull–Rom interpolating splines are constructed as piecewise cubic Hermite interpolating polynomials. In a general interpolation problem, the tangent is not provided and must instead be approximated. To construct the segment connecting the points $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ approximate tangent vectors at those points are needed; call them $\vec{\mathbf{v}}_k^+$ and $\vec{\mathbf{v}}_{k+1}^-$. These, in turn, require an approximation $\tilde{\mathbf{v}}_k$ to $d\mathbf{x}/dt|_{t=t_k}$. We discuss two different approximations; the first based on three-point centered-differences and the second based on five-point centered-differences.

In the first case, under the uniform parameterization $t_k = k\Delta$ this is just $\tilde{\mathbf{v}}_k = (\mathbf{x}_{k+1} - \mathbf{x}_{k-1})/(2\Delta)$, but for general parameterization, it is

$$\tilde{\mathbf{v}}_k = \frac{\Delta_k^2(\mathbf{x}_{k+1} - \mathbf{x}_k) + \Delta_{k+1}^2(\mathbf{x}_k - \mathbf{x}_{k-1})}{\Delta_{k+1}\Delta_k(\Delta_{k+1} + \Delta_k)}, \quad k = 1, \ldots, n-1$$

where $\Delta_k = t_k - t_{k-1}$. Since Hermite interpolating polynomials are not affine invariant with respect to t, this must be scaled by the length of the interval in order to give the right tangent vector to interpolate on $[t_{k-1}, t_k]$. This gives formulas

$$\vec{\mathbf{v}}_k^+ = \Delta_k \tilde{\mathbf{v}}_k \quad \text{and} \quad \vec{\mathbf{v}}_{k+1}^- = \Delta_k \tilde{\mathbf{v}}_{k+1}$$

for the tangent vectors at the left and right end points of the interior points $k = 1, \ldots, n-1$. In the formulas for the tangent vectors $\vec{\mathbf{v}}_0^+$ and $\vec{\mathbf{v}}_n^-$ at the endpoints, the centered difference formula is replaced with a three-point one-sided difference formula, which will, on average, give twice the approximation error as the centered-difference formula.

The second approach, based on five-point centered-differences gives

$$\tilde{\mathbf{v}}_k = \frac{-\mathbf{x}_{k+2} + 8\mathbf{x}_{k+1} - 8\mathbf{x}_{k-1} + \mathbf{x}_{k-2}}{12\Delta_k}, \quad k = 2, \ldots, n-2,$$

under the uniform parameterization. The formula for $\tilde{\mathbf{v}}_k$ under the nonuniform parameterization is of course more complex; see [Gamet *et al.*, 1999]. To derive formulas for the tangent vectors $\vec{\mathbf{v}}_0^+$ and $\vec{\mathbf{v}}_n^-$ at the endpoints, the centered difference formulas are replaced by one-side differences. Similarly, to derive formulas for the tangent vectors $\vec{\mathbf{v}}_1^-$ and $\vec{\mathbf{v}}_1^+$ at the second point, and the tangent vectors $\vec{\mathbf{v}}_{n-1}^-$ and $\vec{\mathbf{v}}_{n-1}^+$ at the second point from the end, the centered difference formulas are replaced by asymmetric differences.

Using (A.7) to express Hermite interpolating polynomials as cubic Bézier curves, we consider Catmull–Rom interpolating splines as composite Bézier curves. The way that the control points of each Bézier segment are constructed guarantees the C^1 continuity of Catmull–Rom splines.

Appendix B

Software

The authors have written a small set of MATLAB programs that implement the methods described in this article. These are available from the first author's website http://web.njit.edu/~goodman/roy/Numerics.html or by searching for "unstable manifold" on the MATLAB File Exchange at http://www.mathworks.com/matlabcentral/fileexchange/.