Computing Periodic Orbits and their Bifurcations with Automatic Differentiation *

John Guckenheimer and Brian Meloon
Mathematics Department, Ithaca, NY 14853

September 29, 1999

1 Introduction

This paper examines algorithms for computing periodic orbits of dynamical systems. Our goal is to explore the limits of accuracy that are attainable for these calculations within the pragmatic context of IEEE floating point arithmetic. A previous paper [13] described a global boundary value solver of very high order accuracy that relies upon automatic differentiation. We extend that work here, introducing multiple shooting algorithms that use automatic differentiation and augmenting these to perform direct computation of saddle-node bifurcations. We compare our algorithms with other numerical methods for finding periodic orbits, examining robustness and accuracy.

Robust algorithms facilitate the automated analysis of dynamical systems. Accuracy is necessary to achieve robustness for two reasons. First, many dynamical systems have fine scale structure that makes them difficult to compute. This is especially true of dynamical systems with multiple time scales and systems that are chaotic. Second, defining equations for bifurcations of periodic orbits involve derivatives of the vector field along the orbit. Newton's method applied to the bifurcation equations requires first derivatives of the defining equations and, hence, second derivatives of the periodic orbit equations. Calculation of second derivatives with finite differences layered on imprecise calculation of periodic orbits is hardly a recipe for robustness. Improved algorithms of greater accuracy for computing periodic orbits and the use of automatic differentiation to compute defining equations enhance existing technology for analysis of dynamical systems.

*Research partially supported by the Air Force Office of Scientific Research, the Department of Energy and the National Science Foundation. The hospitality of the Institute for Mathematics and its Applications is gratefully acknowledged.

Our algorithms approach the limits of precision attainable with double precision IEEE-754 arithmetic for computing periodic orbits. The algorithms utilize three strategies that contribute to their accuracy:

- Automatic differentiation is employed to calculate high order derivatives of a vector field and its trajectories.
- The methods emphasize compact parametrizations of function spaces that include high order approximations to the periodic orbits.
- A posteriori error estimates of the accuracy of numerically computed periodic orbits are used for mesh adaptation.

The methods have additional attractive geometric features from both theoretical and geometric perspectives. They utilize directly the geometric objects that are prominent in the theory. These objects can be readily examined and manipulated, and they can be used adaptively to enable the algorithms to respond to changes in the geometry of a periodic orbit during continuation. The algorithms give **dense output**, representations of approximate periodic orbits at all points rather than just at mesh points of a discretization. Constraints are readily imposed upon mesh points, enabling the accurate computation of periodic orbits of piecewise analytic vector fields.

2 Periodic Orbit Algorithms

The equations describing periodic orbits are boundary value problems for a system of ordinary differential equations. Only a small number of algorithms have been widely used for the computation of periodic orbits. These fall into three classes.

1. Numerical integration of initial values in the domain of attraction of a stable periodic orbit γ converge to γ. (Despite the robustness of numerical integration algorithms, we display examples where they are unable to compute stable periodic orbits.)
2. Shooting methods apply root finding algorithms to approximate flow maps computed with numerical integration.
3. Global methods project the differential equations onto spaces of discretized closed curves and solve these projected equations with root finding algorithms

This paper discusses both shooting methods and global methods of high order. Theoretically, the methods themselves are hardly novel. Our principal contributions examine the limit of increasing order in the methods as contrasted with the limit of increasingly fine meshes. This has some resemblance to p refinement in "h-p" methods studied extensively by Babushka and others [2], but we appear to go much farther in our investigation of convergence on fixed

meshes. This section discusses four different algorithms and variants of these. We use a common terminology for all the algorithms that is now described.

A **periodic orbit** of a vector field

$$\dot{x} = f(x),\ f : R^n \to R^n \tag{1}$$

with period T is a trajectory with $x(T) = x(0)$. T is called the **period** of the orbit. As a point set in the phase space, a periodic orbit may be isolated, but time translation gives a one parameter family family of different parametrized curves corresponding to each periodic orbit. Local analysis of a periodic orbit is based upon the formulation of linear variational equations along the periodic orbit:

$$\dot{\xi} = Df_{x(t)}\xi$$

Taking ξ as a matrix variable, the time T solution of the variational equation with identity initial condition at $t = 0$ is called the **monodromy matrix** of the periodic orbit. The eigenvalues of the monodromy matrix are independent of coordinate system and the parameterization of the periodic orbit. There is always an eigenvalue one with eigenvector tangent to the periodic orbit. A periodic orbit is **elementary** if one is a simple eigenvalue of the monodromy matrix. Elementary periodic orbits are isolated in the sense described above, and they vary smoothly with parameters. We shall denote the flow of the system of equations by $\phi_t(x)$: ϕ_t is the map that advances points t time units along their trajectories.

Periodic orbits are also studied by introduction of **return maps**. A cross-section Σ to f is a codimension one submanifold of R^n with the property that f is never tangent to Σ. The return map $\theta : \Sigma \to \Sigma$ is defined by $\theta(y) = y(s)$ with $y(s)$ the first point of the trajectory with initial condition y that lies in Σ. Periodic orbits have cross-sections whose return maps have fixed points at the intersection of the cross-section with the periodic orbit. The Jacobian derivative of the return map at such a fixed point is closely related to the monodromy of the periodic orbit. In particular, the Jacobian of the return map can be obtained as a quotient of the monodromy map by projecting out the flow direction. The fixed point of a return map for an elementary periodic orbit is an isolated, regular fixed point of the return map. This observation can be used as the basis for simple shooting algorithms for computing the periodic orbits.

We assume throughout this paper that $f(x)$ is analytic. Periodic orbits of analytic vector fields are analytic, but spaces of functions that are discontinuous and/or piecewise smooth underlie algorithms that compute approximate periodic orbits. Since computational representations of periodic orbits only use finite sets of data, we work with finite dimensional function spaces and sequences of such spaces, indexed by a degree. The spaces are parametrized by discrete closed curves, defined below. Each discrete closed curve is used to parametrize a sequence of function spaces of fixed dimension that provide increasingly good approximations to periodic orbits. Different algorithms use different function spaces.

The data structure that we use to represent an approximate periodic orbit is a discrete closed curve. A **discrete curve** $\delta = [(t_0, x_0), (t_1, x_1) \cdots (t_N, x_N)]$ is defined as a set of times $t_0 < t_1 < \cdots < t_N$ and points $x_0, x_1, \cdots, x_N \in R^n$ associated to these times. If $x_0 = x_N$,

then we say that the discrete curve is **closed** and has period $t_N - t_0$. The $(n+1)(N+1)$ dimensional space of discrete curves with $N+1$ points will be denoted $D_{(n,N)}$. The discrete closed curves comprise a linear subspace $D^c_{(n,N)}$ of $D_{(n,N)}$ of dimension $N(n+1)+1$ and codimension n. Each periodic orbit is represented by an $N+1$ dimensional family of discrete closed curves, coming from the selection of different points on the periodic orbit (N degrees of freedom) and time translation. In formulating boundary value methods, we seek systems of defining equations that are square and regular. This can be done by either restricting the domain of the defining equations to a subspace of discrete closed curves which contains a (locally) unique representative of each periodic orbit, or by adjoining additional equations that select a unique discrete closed curve representing the periodic orbit we seek. We have experimented with both strategies. We always remove the degree of freedom due to time translation by restriction to the subspace $t_0 = 0$. Restriction to a subspace that eliminates the degrees of freedom due to moving points on the periodic orbit results in smaller systems of defining equations, but can make the computation of the defining equations more difficult. For example, one strategy for selecting a subspace of discrete closed curves is to select N cross-sections Σ_i to the periodic orbit and restrict x_i to lie in Σ_i. This requires that we define coordinate systems for the Σ_i and incorporate transformations between these coordinates and Euclidean coordinates in our algorithms. Note that as the system size increases, the increase in the number of defining equations, from nN vs. $(n+1)N$, becomes relatively small. Abstractly, the difference between the two strategies can be expressed as the difference between solving a triangular system of equations

$$\begin{aligned} g(x,y) &= 0 \\ h(y) &= 0 \end{aligned}$$

by elimination, first solving $h(y) = 0$ for y and then substituting this value into $g(x,y) = 0$, as contrasted with solving the two dimensional system simultaneously for x and y. Previous work on computing periodic orbits has used elimination methods, but we have found in the canard example described below that augmenting the system of defining equations has produced much better results.

There are tw0 different choices of subspaces of $D^c_{(n,N)}$ that we use to fix the location of points along a periodic orbit. The first is to use cross-sections to the vector field as described above. Cross-sections are hyperplanes of R^n with the property that f is never tangent to Σ_i in the region of interest, defined in the algorithms as hyperplanes perpendicular to the vector field at a point. The subspace $D^s_{(n,N)}$ of $D^c_{(n,N)}$ with $t_0 = 0$ and $x_i \in \Sigma_i$ has an isolated point of intersection with an elementary periodic orbit. The methods most often employed in previous work on computing periodic orbits attempt as much as possible to fix the times $t_0 < t_1 < \cdots < t_N$ of the points $x_0, x_1, \cdots, x_{N-1}$ in the discrete closed curve. Fixing all of the times does not work because the period $t_N - t_0$ of the orbit is not known and must be obtained as part of the solution of the defining equations. There is one degree of freedom that must be retained in allowing the times to vary. This is done typically by either fixing $t_0 < t_1 < \cdots < t_{N-1}$ and allowing t_N to vary, or by fixing the ratios $(t_{i+1} - t_i)/(t_i - t_{i-1})$ for $0 < i < N$. One additional constraint, called a phase condition must be imposed on the

points x_i that eliminates the degree of freedom coming from moving all of the points along the flow simultaneously by time τ and replacing t_i by $t_i - \tau$. Two choices of phase condition that have been employed are to restrict x_0 to lie on a cross-section, and to fix the value of a scalar integral on the periodic orbit.

The next step in defining equations for approximate periodic orbits is to map the spaces of discrete curves to function spaces. Let $\mathcal{C}$ be the space of piecewise smooth curves $g : R \to R^n$ defined by the following properties: $g \in \mathcal{C}$ if there are $u_0 < u_1 < \cdots < u_l$ such that g is analytic on the intervals (u_i, u_{i+1}) and has a C^∞ extension to the closed interval $[u_i, u_{i+1}]$. We consider nonlinear maps $E : D^c_{n,N} \to \mathcal{C}$ with the property that the image functions are analytic on the intervals (t_i, t_{i+1}), $0 \le i < N$. Note that E is allowed to depend upon the vector field f.

A map $E : D^c_{n,N} \to \mathcal{C}$ pulls back the periodic orbit equations from $\mathcal{C}$ to $D^c_{(n,N)}$. These equations give a horrendously overdetermined system on $D^c_{n,N}$, an infinite set of nonlinear equations on a finite dimensional space. Therefore, a finite set of nN equations is selected from the periodic orbit equations. The choice of the map E together with the definition of a regular set of defining equations, perhaps on a subspace $D^s_{(n,N)}$ of $D^c_{(n,N)}$, determines the boundary value solver. We give here a list of four different solvers, specifying the approximate periodic orbit equations for each.

- **Forward Multiple Shooting:** Let $\delta = [(t_0, x_0), \cdots, (t_N, x_N)]$ be a discrete closed curve. The map E assigns to each time interval (t_i, t_{i+1}) of δ the degree d Taylor polynomial p_i of the trajectory with $x(t_i) = x_i$. Since polynomials are analytic on the entire line, this associates a piecewise analytic function in $\mathcal{C}$ to δ. The approximate periodic orbit equations are now defined by

$$P(\delta) = (p_0(t_1) - x_1, \cdots, p_{N-1}(t_N) - x_N)$$

  P vanishes on discrete closed curves δ for which $E(\delta)$ is continuous. It is evident that P is smooth as a map from $R^{(n+1)N+1}$ to R^{nN} since the degree d Taylor series of a trajectory of an analytic vector field depends smoothly on the initial point. The approximate periodic orbit equations P restricted to $D^s_{(n,N)}$ are a set of nN equations in nN variables. Here $D^s_{(n,N)}$ can be the subspace of $D^c_{(n,N)}$ defined via cross-sections and $t_0 = 0$, or by fixing N times and a phase condition. One can also fix $t_0 = 0$ and augment the periodic orbit equations with a set of N additional equations. In the theorem proved below we describe several ways of choosing N augmenting linear equations so that the resulting set of $(n+1)N$ equations in the variables $(x_0, \ldots, x_{N-1}; t_1, \ldots, t_N)$ are regular.

- **Symmetric Multiple Shooting:** Symmetric multiple shooting is a small modification of the multiple shooting method described above that makes the method time reversible. Let $\delta = [(t_0, x_0), \cdots, (t_N, x_N)]$ be a discrete closed curve. We set $u_0 = t_0 - (t_N - t_{N-1})/2$ and $u_i = (t_i + t_{i-1})/2$, $0 < i \le N$. The map E then assigns to each time interval (u_i, u_{i+1}) the degree d Taylor polynomial p_i of the trajectory with $x(t_i) = x_i$. We define the map P by

$$P(\delta) = (p_0(u_0) - p_{N-1}(u_N), p_1(u_1) - p_0(u_1), \cdots, p_{N-1}(u_{N-1}) - p_{N-2}(u_{N-1}))$$

Finally, P is restricted to $D^s_{n,N}$ to obtain a set of nN equations in nN variables, or the set of defining equations is augmented with N additional equations. In implementations of these algorithms, we replace the variables t_i by the equivalent set of variables $t_i - t_{i-1}$ representing the time increment of each mesh interval.

- **Hermite Interpolation:** Let $\delta = [(t_0, x_0), \cdots, (t_N, x_N)]$ be a discrete closed curve. The map E assigns to each time interval (t_i, t_{i+1}) of δ the polynomial p_i of degree $2d+1$ that has the same Taylor series as the trajectories with $x(t_i) = x_i$ and $x(t_{i+1}) = x_{i+1}$. The image of E lies in the set of piecewise analytic functions that are continuous and d times differentiable. We define the function P by

$$P(\delta) = (e_0, \cdots, e_{N-1})$$

with $e_i = \dot{p}_i((t_i + t_{i+1})/2) - f(p_i((t_i + t_{i+1})/2))$. P is restricted to $D^s_{n,N}$ to obtain nN equations in nN variables.

- **Smooth Weighting:** Let $\delta = [(t_0, x_0), \cdots, (t_N, x_N)]$ be a discrete closed curve, and let $\beta : [0, 1] \to [0, 1]$ be a monotonic C^∞ function that is flat at its endpoints ($\beta^{(k)}(u) = 0$ for all $k > 0$) and has $\beta(0) = 0$, $\beta(1) = 1$. We assume that β is analytic in $(0, 1)$ with positive derivative. Let p_i be the degree d Taylor polynomial of the trajectory with $x(t_i) = x_i$. To each interval (t_i, t_{i+1}), we assign the C^∞ function

$$g_i(t) = (1 - \beta((t - t_i)/(t_{i+1} - t_i)))p_i(t) + \beta((t - t_i)/(t_{i+1} - t_i))p_{i+1}(t)$$

This defines a C^∞ closed curve that interpolates the degree d Taylor series expansions at the mesh points. We define the function P by

$$P(\delta) = (e_0, \cdots, e_{N-1})$$

with $e_i = \dot{g}_i((t_i + t_{i+1})/2) - f(g_i((t_i + t_{i+1})/2))$. P is restricted to $D^s_{n,N}$ to obtain nN equations in nN variables.

Each of these four methods yields a system of equations defined by a map P that depends on the same number of equations as variables. Note that there is an inherent difference between the first two algorithms and the last two. The first two are multiple shooting algorithms that use spaces of discontinuous functions and their maps P do not depend directly upon evaluating the vector field. The approximation to the periodic orbit is implicit in the choice of the function space and depends upon the accuracy of the trajectory segments. The last two solvers are global methods defined on smooth curves (the degree of smoothness of the Hermite interpolations depends on d) and their equations P evaluate both the vector field at selected points and the tangent vectors to curves in the function space.

We want convergence proofs and error estimates for these algorithms in terms of the vector field, characteristics of the periodic orbit and algorithmic parameters. There are two types of limits that are of interest: increasing degree d of Taylor expansions and increasing

mesh fineness. For fixed degree of the Taylor polynomials and increasingly fine meshes, the algorithms are variants of standard shooting and collocation algorithms. Therefore, we focus our mathematical discussion on the limit of increasing degree with a fixed mesh. This provides mathematical foundations for algorithms that seek to compute periodic orbits using polynomials of high degree with coarse meshes.

The theorems stated below depend upon the geometry of the periodic orbits that are being approximated. If periodic orbits are not elementary, there is little hope that the projected equations P will be regular. Thus, our convergence results focus on the case of elementary periodic orbits. We assume that all discrete closed curves $\delta = [(t_0, x_0), (t_1, x_1) \cdots (t_N, x_N)]$ have the property that the radius of convergence of the trajectories through the points x_i are larger than $t_i - t_{i-1}$ and $t_{i+1} - t_i$. Within a compact subset of R^n, there is a number $D > 0$ such that this property will be satisfied if $|t_i - t_{i-1}| < D$ for each i. In particular, this assumption implies that the Taylor polynomials of the trajectories converge uniformly to the trajectories with increasing degree throughout each mesh interval. The assumption that meshes are sufficiently fine that each mesh interval is contained in the radius of convergence of the trajectories at its endpoints will be maintained throughout this paper.

Theorem Let γ be a periodic orbit of the vector field $\dot{x} = f(x)$ such that 1 is a simple eigenvalue of its monodromy matrix. Let $D \subset R^{(n+1)(N+1)}$ be the $(n+1)(N)$ dimensional subspace with coordinates $(x_0, \ldots, x_N; t_0 \ldots t_N)$ defined by $x_0 = x_N$ and $t_0 = 0$. For any $\epsilon > 0$, there are

- a neighborhood U of γ
- $\tau > 0$
- integers d, N

so that the system of equations

$$|x_{i+1} - p_i(t_{i+1})| = 0,\ i = 0, \ldots, N-1$$

has a smooth N parameter family of solutions in D with

- p_i the Taylor polynomial of degree d for the trajectory taking the value x_i at t_i
- $0 < s_i = t_{i+1} - t_i < \tau$
- $|x_{i+1} - \phi_{s_i}(x_i)| < \epsilon$

The Jacobian of the system of equations has maximal rank nN.

Proof: Select a neighborhood U of γ and τ so that the Taylor series of trajectories with initial point in U have radius of convergence larger than τ. Next, select a set of N points $y_0, \ldots, y_{N-1}, y_N = y_0$ on γ so that $y_{i+1} = \phi_{u_i}(y_i)$ with $u_i < \tau$. Set

$$r_i = \sum_{i=0}^{j} u_j$$

Consider the map $F_f^d : D \to R^{nN}$ defined by

$$\begin{gathered} p_0(r_0) - x_1 \\ p_1(r_1) - x_2 \\ \vdots \\ p_{N-2}(r_{N-2}) - x_{N-1} \\ p_{N-1}(r_{N-1}) - x_0 \end{gathered}$$

as an approximation of the map $F_f^\infty : D \to R^{nN}$

$$\begin{gathered} \phi_{s_0}(x_0) - x_1 \\ \phi_{s_1}(x_1) - x_2 \\ \vdots \\ \phi_{s_{N-2}}(x_{N-2}) - x_{N-1} \\ \phi_{s_{N-1}}(x_{N-1}) - x_0 \end{gathered}$$

The map F_f^d depends upon the degree d of the Taylor polynomials and converges to the map F_f^∞ as $d \to \infty$. (Since the maps F_f^∞ and F_f^d are defined on finite dimensional spaces of discrete closed curves, convergence is independent of norms for these spaces.) The map F_f^∞ has an N dimensional manifold S of zeros passing through the discrete closed curve $[(r_0, y_0), \ldots (r_N, y_N)] \in D$. S is obtained by varying the points y_i along γ and making suitable changes in the u_i. If this zero of F_f^∞ is regular, then, for sufficiently large d, F_f^d will have a regular zero that converges to the zero of F_f^∞. Thus, the crux of the proof lies in demonstrating that the rank of DF_f^∞ is nN.

We relate the Jacobian of F_f^∞ to the monodromy of γ. In D, we use coordinates $(x_0, \ldots, x_{N-1}; t_1, \ldots, t_N)$. The Jacobian DF_f^∞ has a block structure

$$DF_f^\infty = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ -I & & & & D_N & & & & & -v_N & v_N \end{pmatrix}$$

with $D_i = D_x(\phi)(x_{i-1}, t_i - t_{i-1})$ and $v_i = f(x_i)$. Note that $D_i(v_{i-1}) = v_i$. The blocks D_i are nonsingular since the flow maps ϕ_t are diffeomorphisms. To determine whether the rank of the matrix DF_f^∞, we compute its LU factorization in the form

$$L = \begin{pmatrix} I & & & & \\ & \cdots & \cdots & & \\ & & & & I \\ L_1 & L_2 & \ldots & L_{n-1} & I \end{pmatrix}$$

$$U = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ & & & & \tilde{D}_N & w_1 & w_2 & \cdots & \cdots & w_{N-1} & w_N \end{pmatrix}$$

This yields a system of equations for the L_i, $\tilde{D}_N$ and w_i:

$$\begin{aligned} L_1 D_1 &= -I \\ L_i D_i - L_{i-1} I &= 0 \\ \tilde{D}_N - L_{N-1} E_{N-1} &= D_N \\ L_{i-1} v_{i-1} - L_i v_i + w_{i-1} &= 0 \\ L_{N-1} v_{N-1} + w_{N-1} &= -v_N \\ w_N &= v_N \end{aligned}$$

where $1 < i < N$. The solutions of this system are

$$\begin{aligned} L_i &= -D_1^{-1} \cdots D_i^{-1} \\ \tilde{D}_N &= D_N - D_1^{-1} \cdots D_{N-1}^{-1} \\ w_i &= 0 \\ w_N &= v_N \end{aligned}$$

with $0 < i < N$. The factor L has rank nN since it is lower triangular with ones along the diagonal. The factor U has rank nN if and only if the rank of $\tilde{D}_N$ is $n-1$ and v_N does not lie in the image of $\tilde{D}_N$. By definition, γ is an elementary periodic orbit if and only 1 is a simple eigenvalue of the monodromy map $D_N \cdots D_1$. Thus, $\tilde{D}_N$ has rank $n-1$ with kernel v_{N-1} if γ is elementary. If $v_N = v_0$ is in the image of $\tilde{D}_N$, then $v_{N-1} = D_{N-1} \cdots D_1(v_0)$ is in the image of $D_{N-1} \cdots D_1 \tilde{D}_N = D_{N-1} \cdots D_1 D_N - I$. But $D_{N-1} \cdots D_1 D_N$ is the monodromy map with initial point y_{N-1} and v_{N-1} is the eigenvector for the simple eigenvector 1, so v_{N-1} is not in the image of $D_{N-1} \cdots D_1 D_N - I$. We conclude that if γ is elementary, then v_N does not lie in the image of $\tilde{D}_N$ and U has rank nN. Thus the system of equations F_f^∞ has maximal rank at a zero formed from N points on an elementary periodic orbit γ. Perturbations of this system also have maximal rank and the manifold of solutions varies smoothly with perturbation. Since the Taylor polynomials of the flow map converge to the flow map on time intervals of length smaller than τ, the conclusions of the theorem will be satisfied for the map F_f^d when the degree d of the Taylor polynomials is sufficiently large. This proves the theorem.

Similar arguments to the one used above prove the convergence of other multiple shooting algorithms, notably the symmetric multiple shooting algorithm with increasing degree.

The zeros of the map F_s^d

$$\begin{array}{c} \phi_{s_1}(x_0) - \phi_{-s_1}(x_1) \\ \phi_{s_2}(x_1) - \phi_{-s_2}(x_2) \\ \vdots \\ \phi_{s_{N-1}}(x_{N-2}) - \phi_{-s_{N-1}}(x_{N-1}) \\ \phi_{s_N}(x_{N-1}) - \phi_{-s_N}(x_0) \end{array}$$

with $s_i = (t_i - t_{i-1})/2$ give discrete closed curves on the periodic orbit γ. The Jacobian of F_s^d has a similar block structure to that for forward multiple shooting:

$$DF_s^d = \begin{pmatrix} D_1 & -E_1 & & & & v_1 & & & & & \\ & D_2 & -E_2 & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -E_{N-1} & & & & -v_{N-1} & v_{N-1} & \\ -E_N & & & & D_N & & & & & -v_N & v_N \end{pmatrix}$$

with D_i the Jacobian of ϕ_{s_i} at x_{i-1}, E_i the Jacobian of ϕ_{-s_i} at x_i and $v_i = f(\phi_{-s_i}(x_i))$. Multiplying DF_s^∞ on the right by the block diagonal matrix with entries

$$E_2^{-1}, E_3^{-1}, \ldots, E_N^{-1}, E_1^{-1}, E_2^{-1}, E_3^{-1}, \ldots, E_N^{-1}, E_1^{-1}$$

gives the same matrix as DF_f^∞ above.

Restriction of F_f^∞ or F_f^d to a subspace $L \subset R^{(n+1)N}$ is equivalent to augmenting these maps with an additional linear map whose kernel gives the subspace L. There are three choices of subspace we have used in our numerical work. The method used in AUTO is to fix a single linear constraint of the phase space variables x_i (called a *phase condition*) and to fix the times t_i up to a scale factor that yields the period of the periodic orbit. The second method is to fix cross-sections Σ_i orthogonal to the vector field at x_i and constrain x_i to vary in Σ_i. The Jacobian DF_f^∞ for the augmented map is

$$J_c = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ -I & & & & D_N & & & & & -v_N & v_N \\ v_0^t & & & & & & & & & & \\ & v_1^t & & & & & & & & & \\ & \cdots & \cdots & & & & & & & & \\ & & \cdots & \cdots & & & & & & & \\ & & & v_{N-2}^t & & & & & & & \\ & & & & v_{N-1}^t & & & & & & \end{pmatrix}$$

If γ is an elementary periodic orbit, then J_c is non-singular. This is proved as follows. If $J_c(y_0^t, \ldots, y_{N-1}^t, u_1, \ldots, u_N) = 0$ then

$$v_i \cdot y_i = 0$$
$$D_i y_i - y_{i+1} = (u_i - u_{i-1}) v_{i+1}$$

for $i = 0, \ldots, N-1$ with $u_{-1} = 0$. Let π_i be the projection of R^n onto Σ_i. Then $\pi_i D_i y_{i-1} - \pi_i y_i = 0$. If $y_0 \neq 0$, it is an eigenvector of $\pi_N D_N \ldots \pi_1 D_1$ with eigenvalue 1. But $\pi_N D_N \ldots \pi_1 D_1$ is the Jacobian of the return map of Σ_0, and 1 is not an eigenvalue. We conclude that $y_0 = 0$ and consequently $\pi_i y_i = 0$ for all i. Together with the equations $v_i \cdot y_i = 0$, this implies $y_i = 0$. Finally, we conclude that $(u_i - u_{i-1}) v_{i+1} = 0$, implying $u_i = 0$. Thus J_c is non-singular.

The third way we restrict F_f^∞ to a subspace appears to be novel, but natural. We observe that if $[(t_0, x_0), \ldots, (t_N, x_N)]$ is a discrete closed curve lying in γ, then so is the curve that replaces (t_i, x_i) by $(\phi_h(x_i), t_i + h)$. We restrict to the subspace orthogonal to these curves for $i = 1, \ldots, N-1$. For $i = 0$, we preserve the constraint $t_0 = 0$ by restricting to the orthogonal complement to the vector $(v_0, \ldots, 0; -1, \ldots -1, 0)$ corresponding to $x_0 \to \phi_h(x_0)$ and $t_i \to t_i - h$ for $i = 1, \ldots, N-1$. This gives a restricted subspace that is orthogonal to the manifold of solutions of $F_f^\infty = 0$.

The theory underlying the Hermite interpolation algorithm is somewhat more complex than the other algorithms described here because the Hermite polynomials on a fixed mesh interval approach a discontinuous function with increasing degree. A particular example illustrates this. Consider polynomials p_d of degree $2d+1$ with the properties that $p_d(-1) = 0$, $p_d(1) = 1$ and $p_d^{(j)}(\pm 1) = 0$ for $1 < j \leq d$. There is an explicit formula for p_d, namely $p_d = (\frac{1+x}{2})^{(d+1)} q_d$ where q_d is the degree d Taylor polynomial of $(\frac{1+x}{2})^{-(d+1)}$ at $x = 1$. Figure 1 shows a plot of p_{11}. The polynomials p_d are Hermite interpolants between the points $(x_0, t_0) = (0, -1)$ and $(x_1, t_1) = (1, 1)$ for the trivial vector field $\dot{x} = 0$ on R. As d increases, the p_d converge to constant functions 0 and 1 on $[-1, 0)$ and $(0, 1]$ with a "transition" layer of length comparable to $1/\sqrt{d}$. This behavior is typical:

Theorem: (Shen and Strang [21, 22]) Let g and h be two functions analytic on the interval $[-1, 1]$, let g_d and h_d be the degree d Taylor polynomials of g and h at -1 and 1, and let p_d be the polynomial of degree $2d+1$ whose Taylor polynomials of degree d at -1 and 1 agree with those of g and h respectively. Assume that the Taylor series of g at -1 and h at 1 have radius of convergence larger than 1. As d increases

$$p_d(x) = \begin{cases} g(x) + O(\frac{|1-x^2|^d}{\sqrt{d}|x|}), & x < 0 \\ h(x) + O(\frac{|1-x^2|^d}{\sqrt{d}|x|}), & x > 0 \end{cases}$$

Remark: The values $p_d(0)$ and $p_d'(0)$ are both linear combinations of the Taylor series coefficients of g and h. The magnitude of $p_d'(0)$ is comparable to $|g(0) - h(0)|\sqrt{d}$.

The Hermite interpolation algorithm constructs interpolating polynomials on each interval of a discrete closed curve from the Taylor series coefficients of trajectories at mesh

Figure 1: The Hermite interpolating polynomial of degree 23 connecting constant functions 0 and 1 on the interval $[-1, 1]$. The interpolating polynomial is almost flat at the ends of the interval with a monotonic transition layer near its center.

points. The system of defining equations for the algorithm comes from collocation in each mesh interval: $p'(\tau) - f(p(\tau)) = 0$ with p the Hermite interpolating polynomial and τ the midpoint of the interval. To obtain a limiting set of equations for a discrete closed curve lying in a periodic orbit, these equations need to be scaled by $\sqrt{d}$ with increasing d.

Theorem: Let γ be a periodic orbit of the vector field $\dot{x} = f(x)$ such that 1 is a simple eigenvalue of its monodromy matrix. If $\delta = [(t_0, x_0), (t_1, x_1) \cdots (t_N, x_N)]$ is a discrete closed curve, denote by p_i the degree $2d+1$ Hermite interpolation of the Taylor series of the trajectory segment on mesh interval i of δ, by τ_i the midpoint (in time) of mesh interval i and set $G_i = (p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$. If d is sufficiently large, δ is close to γ and the mesh of δ is sufficiently fine, then the system of equations $G_i = 0$ with the x_i constrained to lie on a set of cross-sections to γ form a regular system of equations whose solutions converge uniformly to γ as $d \to \infty$.

We give only a brief outline for the proof of this theorem. There are two aspects of the proof; namely, the analysis outlined above of how the Hermite polynomials approximate trajectory segments and demonstration that the Jacobian of the system of equations is a regular, square matrix. Denoting the degree d Taylor series approximations to γ at the points of δ by q_i, the magnitude of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ is comparable to $|q_{i+1}(\tau_i) - q_i(\tau_i)|$. This is the leading order term of the asymptotic expansion of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ in d. Since we assume that each mesh interval is contained within the domain of convergence of the Taylor series for the trajectories at its endpoints, there is a $0 < \rho < 1$ so that $|q_i(\tau_i) - \phi(x_i, \tau_i - t_i)|$ decreases at a rate faster than ρ^d. Therefore, $|\phi(x_{i+1}, \tau_i - t_{i+1}) - \phi(x_i, \tau_i - t_i)|$ decreases at a rate faster than ρ^d as d increases, implying that the solutions of $G_i = 0$ converge uniformly to points of γ as $d \to \infty$. The Jacobian has a block matrix structure of the same kind as a symmetric multiple shooting algorithm since the leading order term of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ approaches $|\phi(x_{i+1}, \tau_i - t_{i+1}) - \phi(x_i, \tau_i - t_i)|$. The regularity of the system of equations $G_i = 0$ follows from the regularity of the Jacobian of symmetric multiple shooting with phase points restricted to a set of cross-sections to the periodic orbit.

3 Continuation and Bifurcation

Introduction of parameters prompts two extensions to the algorithms that we have described above, namely

1. computing families of solutions with changing parameters, and
2. computing bifurcations: parameter values at which the stability properties of the periodic orbits change.

Continuation methods, algorithms for computing curves of solutions to a system of k equations in $k+1$ variables have been extensively studied. These can be used in a straightforward way with the periodic orbit algorithms described above. We give a brief general description of continuation methods. If a system of equations $F(x_1, \ldots, x_{k+1}) = 0$ has maximal rank, the implicit function theorem implies that the solution is a smooth curve whose tangent direction is the kernel of DF. The solution curve can be parametrized either by arc length or, locally, by a suitable coordinate x_i. Points on the solution curve can be calculated by simple predictor-corrector methods using an Euler step along the curve as a predictor and a Newton iteration in a hyperplane of R^{k+1} as a corrector. Methods for adapting the step length along the solution curves have been incorporated into computer packages for continuation [**?**, 5, 17]. For computing equilibria or periodic orbits of a system of ordinary differential equations, one of the coordinates in the system of equations $F(x_1, \ldots, x_{k+1}) = 0$ is a system parameter, called the **active parameter**.

Local bifurcations of periodic orbits involve a change of stability along a smooth family of periodic orbits or the collapse of a family of periodic orbits at a point of equilibrium Hopf bifurcation [14]. Global bifurcations of periodic orbits are associated with the period of an orbit becoming unbounded, either by approaching an orbit homoclinic to a saddle or by approaching a saddle-node in a cycle bifurcation [14]. Here we consider only local bifurcations of periodic orbits; i.e, those involving a change of stability in the periodic orbit. The defining equations for such bifurcations can be formulated most easily in terms of the Jacobian J of the return map for an orbit. A codimension one bifurcation occurs if J has an eigenvalue of modulus 1 and suitable non-degeneracy conditions are satisfied. There are three cases: saddle-node bifurcation with eigenvalue 1, period doubling (also called flip) bifurcation with eigenvalue -1, and Hopf (also called torus) bifurcation in the case of a complex pair of eigenvalues of modulus 1.

The principal algorithmic question in computing bifurcations is the formulation of defining equations. Since the defining equations are expressed in terms of the Jacobian J of the return map or the monodromy of the periodic orbit, it is important to compute these matrices accurately. Our work seeks to connect the accurate calculation of periodic orbits and their return maps with the best algorithms from linear algebra for computing defining equations for bifurcation. Where possible, we seek to reduce the defining equations to a single equation expressing the singularity of a matrix. Theoretically, this is easily done for eigenvalues ± 1: the determinants of the matrices $J \mp I$ are defining equations for the saddle-node and period doubling bifurcations. For Hopf bifurcation, the tensor product matrix $J \otimes J$ acting on skew-symmetric tensors has eigenvalues that are pairwise products of distinct eigenvalues of J. Therefore, the determinant of the matrix $J \otimes J - I \otimes I$ gives a defining equation for Hopf bifurcation.

While determinants of matrices give defining equations for determining whether a matrix A is singular, there are better methods. The smallest singular value of A is a better conditioned and more reliable measure than $\det A$ of the distance of A from the set of singular matrices. Bordered matrix methods [11] provide an efficient way to compute a quantity proportional to the smallest singular value using Gaussian elimination with partial pivoting.

If A has corank one, then for generic vectors B and C, the bordered matrix

$$\begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix}$$

is nonsingular. In this case, the equations

$$\begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix} \begin{pmatrix} V \\ H \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$(W^t \quad H) \begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix} = (0 \quad 1)$$

have unique solutions V, W, H with H a scalar that is proportional to the smallest singular value of A [11]. For computing saddle-node bifurcations, we shall chose matrices A related to the Jacobian DF_*^d of our system of periodic orbit equations F_*^d. It has a block structure, and is nonsingular if and only if 1 is not an eigenvalue for the return map of γ. Therefore, we border DF_*^d and add the equation $H = 0$ as the augmenting defining saddle-node bifurcation.

In applying Newton's method to the system

$$\begin{pmatrix} F_*^d \\ H \end{pmatrix} = 0$$

we need partial derivatives of F_*^d and H with respect to x_i, t_i and λ. For $z \in \{t_i, x_i, \lambda\}$,

$$H_z = -W^t (\frac{\partial DF_*^d}{\partial z} V$$

Therefore, computation of the partial derivatives of H is reducible to computation of second derivatives of DF_*^d.

4 Implementations and Adaptation

This section describes implementations of the algorithms described in previous sections. Matlab 5.2 [19] was used to program the higher level parts of the algorithms, and ADOL-C 1.7 [12] was to used to compute the Taylor series expansions of trajectories with automatic differentiation. We modified a function in ADOL-C that computes the derivatives of the Taylor series coefficients with respect to the phase space variables so that it also computes derivatives of the Taylor series coefficients with respect to parameters. Summing the derivatives of these Taylor series coefficients gives the Jacobian of the flow map and the derivative of the flow map with respect to parameters. We have investigated several variations of the algorithms but have not tested them enough to form definitive conclusions about which variants are recommended in which circumstances. Indeed, as with numerical integration of initial value problems, individual preferences are likely to persist in the use of different algorithms. Here we describe three algorithms, each of which appears to work well on at least some of the test problems described in the next section.

4.1 Forward multiple shooting with adaptive step lengths

The first algorithm whose implementation we describe is forward multiple shooting. The input for the algorithm is $\delta = [(t_0, x_0), (t_1, x_1) \cdots (t_N, x_N)]$, a discrete closed curve that approximates a periodic orbit. The desired output is a new discrete closed curve $\pi = [(s_0, y_0), (s_1, y_1) \cdots (s_N, y_N)]$ that approximates the periodic orbit with high precision. This is achieved by computing the approximate flow map $p_{t_{i+1}-t_i}(x_i)$ using the Taylor polynomial of ϕ at x_i and forming the map

$$F_f^d(\delta) = (e_0, ...e_{N-1}), \qquad e_i = p_{t_{i+1}-t_i}(x_i) - x_{i+1}$$

that evaluates the difference between a point of δ and the flow from the previous point on the discrete closed curve. Restricting to the subspace defined by $x_0 = x_N$ and $t_0 = 0$, there are still N more independent variables than equations in F_f^d. We want to apply Newton's method to F_f^d. This is done by requiring that the Newton updates are orthogonal to the N vectors $f(x_i)\partial x_i + \partial t_i$ for $0 < i < N$ and $f(x_0)\partial x_0 - \sum_{i=1}^{N-1} \partial t_i$. Concretely, we use coordinates $(x_0, \ldots, x_{N-1}, t_1, \ldots, t_N)$ and add N rows to DF_f^d to obtain the matrix

$$J_o = \begin{pmatrix}
D_1 & -I & & & & v_1 & & & & & \\
 & D_2 & -I & & & -v_2 & v_2 & & & & \\
 & \cdots & \cdots & & & & \cdots & \cdots & & & \\
 & & \cdots & \cdots & & & & \cdots & \cdots & & \\
 & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\
-I & & & & D_N & & & & & -v_N & v_N \\
v_0^t & & & & & -1 & -1 & \ldots & -1 & -1 & \\
 & v_1^t & & & & 1 & & & & & \\
 & & \cdots & & & & \cdots & & & & \\
 & & & v_{N-2}^t & & & & & 1 & & \\
 & & & & v_{N-1}^t & & & & & 1 &
\end{pmatrix}$$

We iterate the Newton map that subtracts the vector $J_o^{-1}(e_0, \ldots e_{N-1}, 0, \ldots, 0)$ from $(x_0, \ldots, x_{N-1}, t_1, \ldots, t_N$ to converge to a discrete closed curve that approximates the periodic orbit.

The accuracy of the solution to equations $F_f^d = 0$ as an approximation to the periodic orbit depends upon the accuracy of the time steps of numerical integration. These are computed by evaluation of Taylor polynomials with a fully adaptive step length procedure. The Taylor polynomials themselves and their Jacobians are computed with the programs `forode` and `accode` in the computer package ADOL-C [12]. These automatic differentiation algorithms use no finite difference calculations and typically produce Taylor series of high degree that are close to the numerical precision of IEEE-754 floating point arithmetic. Step lengths are chosen based upon two criteria:

1. the estimated radius of convergence of the Taylor series, and
2. agreement between the tangent vector to the Taylor polynomial and the vector field at the end of the step.

Our implementation of the first criterion assumes that the rate of growth of the computed coefficients a_i of the Taylor series is a good estimate of the radius of convergence of the Taylor series. The quantity $\rho = \sup |a_i|^{-1/i}$ over some range of degrees i is used to estimate the radius of convergence. Assuming that the degree j term in the Taylor series of the trajectory is bounded by ρ^j, we estimate a step length h with the property that the remainder of the Taylor series is smaller than the numerical precision of the computations along the step. If the size of the estimated step is larger than the time to the next mesh point, we reduce it to end at the time of the next mesh point. With this choice of step we hope that the accuracy of the computed step is within unit precision of the floating point arithmetic from the exact value. As an a posteriori check on the accuracy of the numerical trajectory, its tangent vector at the end of the step is compared with the value of the vector field at this point. If the difference between these exceeds a specified (relative or absolute) tolerance, the step size is reduced (we used a factor of 0.7) and the test repeated at the end point of the step of reduced length. If no step larger than a minimum specified step length satisfied the desired bound for the difference between tangent vector and vector field, then the algorithm halted with an error message. The map F_f^d is computed by numerically integrating each mesh point to the time of the next mesh point in this manner. As the numerical integration is done, the Jacobian of the flow is also computed, using the derivatives of the Taylor series coefficients with respect to the phase space variables, as computed with ADOL-C.

The motivation for using multiple shooting rather than single shooting stems largely from situations in which the norm of the flow map along parts of a trajectory become so large that the computation of the flow maps are horribly ill-conditioned. In extreme situations like those illustrated in the canard example of the next section, some trajectory segments within a periodic orbit are so unstable that a numerical integration cannot follow them without extreme floating point precision. We need flow maps from one mesh point to the next to be sufficiently well conditioned that small changes in the initial point of the trajectory produce small changes in its final point. We use a mesh refinement algorithm to produce such flow maps. The algorithm monitors the magnitude of the Jacobian of the flow map from the previous mesh point. When this magnitude exceeds a specified bound, then a new mesh point is inserted. The result is a discrete closed curve with imposed bounds on the magnitude of the Jacobian of the flow map from the beginning to the end of each trajectory segment.

In addition to mesh refinement, we also implement an algorithm for mesh decimation. The strategy which is used is the following. The first point of the mesh is kept. If the last mesh point which is kept has index i, its trajectory is integrated to the time of the next mesh point. If $|p(x_i, t_{i+1} - t_i) - x_{i+1}|$ is larger than a chosen tolerance or the norm of the Jacobian of p is larger than a chosen bound, then the mesh point (t_{i+1}, x_{i+1}) is kept. If neither of these conditions fails, the numerical integration is continued to the first time t_{i+k+1} at which either $|p(x_i, t_{i+k+1} - t_i) - x_{i+k+1}|$ is larger than its tolerance or the Jacobian of p is larger than its bound. The mesh point (t_{i+k}, x_{i+k}) with index $i + k$ is then retained and the numerical integration restarted from x_{i+1}. Finally, (t_N, x_N) is retained so that we have a discrete closed curve.

Our implementation of this forward shooting algorithm is embedded in a simple continuation framework. A parameter λ is designated as the active parameter, and an additional column giving the derivatives of F_f^d with respect to λ is added to the matrix JA, producing a matrix J_a. Computation of these derivatives required modifications to the program `accode` in ADOL-C. As the active parameter is varied, the equations $F_f^d = 0$ have an $N+1$ dimensional manifold of solutions. The curve on the solution manifold that satisfies the constraints imposed above has tangent vector lying in the null space J_a, spanned by the unit vector C. This null space is computed with MATLAB, and the discrete closed curve $[(t_0, x_0) \dots (t_N, x_N); \lambda]$ is augmented by adding a small multiple of C to the last computed discrete closed curve to obtain the initial seed for the next Newton iteration. When computing periodic orbits in the canard family, it is necessary to allow the parameter to vary during the Newton iteration since the amplitude of periodic orbits is extremely sensitive to parameter values. Thus we add one more row to J_a, the transpose of C, and solve for Newton updates of the variables $(x_0, \dots, x_{N-1}, t_1, \dots, t_N, \lambda)$. The code to implement this algorithm on top of ADOL-C is very small. Including continuation of the periodic orbits with a varying parameter, it comprises less than 600 lines of Matlab m-files, and two C++ files to evaluate the vector field and provide the interface between Matlab and ADOL-C used in computing Taylor series of trajectories.

4.2 Hermite Polynomial Interpolation

The Hermite polynomial global algorithm is somewhat more complicated than the multiple shooting algorithm described above for three reasons:

1. The algorithm constrains the phase space variables to lie on cross-sections and uses the times increments $s_i = t_i - ti - 1$, $0 < i \leq N$, as independent variables.

2. The calculation of the Hermite polynomials from the Taylor series at the ends of a mesh interval is more sensitive to round-off errors for high degree interpolation.

3. The defining equations for the approximate periodic orbit and their Jacobians are more complicated than for the multiple shooting algorithms.

The algorithm takes as input a discrete closed curve $\delta = [(t_0, x_0), \cdots, (t_N, x_N)]$ and seeks to compute a new discrete closed curve whose Hermite interpolation is an accurate approximation to a periodic orbit. The core of the algorithm is a Newton iteration to solve the equations $G_i = 0$ defined in our discussion of the theoretical foundations of this algorithm. We compute the Hermite polynomials by translating each mesh interval $[t_i, t_{i+1}]$ to $[-\beta, \beta]$ with $\beta = (t_{i+1} - t_i)/2$ and use a basis for the polynomials of degree $2d + 1$ consisting of the polynomials

$$(1 - (t/\beta)^2)^j \qquad \text{and} \qquad (t/\beta)(1 - (t/\beta)^2)^j,\ 0 \leq j \leq d$$

Using this normalization, the value of the interpolating polynomial and its derivative at the midpoint of the mesh interval is expressed directly as a weighted sum of the coefficients of the

Taylor series at the mesh points. The Taylor series and the derivatives of the Taylor series coefficients with respect to phase space variables are computed using the routines `forode` and `accode` in the ADOL-C package. Cross-section coordinates are used in the Newton iteration. At the beginning of the Newton iteration, the vector field f is evaluated at each mesh point x_i and an orthonormal basis for the subspace Σ_i orthogonal to x_i is computed as the last $n-1$ columns of the Q factor of the QR decomposition of the matrix $[f(x_i)\, I]$. Coordinates with respect to this basis are used to parametrize Σ_i. The Jacobian of the G_i is computed with respect to these coordinates at x_i and x_{i+1}, as well as with respect to changes in s_i, the length of mesh interval i.

The goal of the Hermite interpolation algorithm is to produce a curve p with $E(x) = |p'(x) - f(p(x))|$ smaller than a specified bound. Following convergence of the Newton iteration, we expect to meet this criterion at the endpoints and midpoints of the mesh intervals. To estimate whether the bound is satisfied uniformly along p, we evaluate E at additional points equally spaced in each mesh interval. If the desired bound on E is not met, we refine the mesh by adding to the mesh the midpoints of the mesh intervals with the largest values of E. We experimented with different criteria for determining when a mesh interval would be subdivided. The performance of the varied mesh refinement strategies appeared to be comparable to one another. We employed a different mesh strategy in this algorithm than in the forward shooting algorithm described above. Here, mesh points are deleted if the the apparent errors of Hermite interpolating polynomials on the neighboring mesh intervals lie below a specified bound. We do not think that this is an optimal decimation strategy, but it was simple to implement and its performance was satisfactory with the test problems we investigated.

4.3 Saddle-node bifurcations with multiple shooting

The next algorithm we describe is a symmetric multiple shooting algorithm for finding saddle-node bifurcations of periodic orbits. Similar modifications can be made to any of our methods, but the multiple shooting algorithms are simpler than the global methods. The algorithm takes a discrete closed curve that approximates a periodic orbit and a starting parameter λ_0 as input. The desired output is a new discrete closed curve $[(t_0, x_0), (t_1, x_1) \cdots (t_N, x_N]$ and a new parameter λ which approximates with high accuracy a periodic orbit undergoing a saddle-node bifurcation.

The method is implemented with points constrained to lie in fixed cross-sections, so we define a set of cross-sections Σ_i orthogonal to the vector field at the current mesh and introduce coordinates $w_i \in R^{n-1}$ for Σ_i. Each (half) mesh interval is traversed in a single step of length $s_i = (t_{i+1} - t_i)/2$.

We compute the approximate flow maps $p_{s_i}(w_i)$ and $p_{-s_i}(w_{i+1})$ using the degree d Taylor polynomials of ϕ at w_i and w_{i+1}. These are used to form the map $F = D_s^d$

$$F(s_0, w_0, \ldots, s_{N-1}, w_{N-1}, \lambda) = (e_0, ...e_{N-1}), \qquad e_i = p_{s_i}(w_i) - p_{-s_i}(w_{i+1})$$

that evaluates the difference between the end of the trajectory segment and the next mesh point. The Jacobian DF is non-singular if and only if the periodic orbit is regular. Thus the singularity of DF determines points of saddle-node bifurcation. We compute the singularity of DF with a bordered matrix computation. Assume that DF has only a single singular value that is close to 0. We can then border DF by a single row and column to form a matrix M whose smallest singular value is far from zero. The last component of the solution v to the matrix equation $Mv = (0, \ldots, 0, 1)^t$ is comparable to the smallest singular value of the matrix DF [9]. Denote this last component of v by $H(s, w, \lambda) = 0$. In our algorithms, H is computed from M by functions that are part of MATLAB.

We next apply Newton's method to the system

$$\begin{pmatrix} F \\ H \end{pmatrix} = 0$$

In doing this, we need to compute the Jacobian of the system, which has the following form:

$$K = \begin{pmatrix} DF & \frac{\partial F}{\partial \lambda} \\ \frac{\partial H}{\partial (x,\delta)} & \frac{\partial H}{\partial \lambda} \end{pmatrix}$$

The computation of DF is the same as it was for the previous section. Given our modifications to ADOL-C, the computation of $\frac{\partial F}{\partial \lambda}$ is also straightforward. Derivatives of H at solutions to $H = 0$ satisfy $H_z = -W^t(DF_z)V$ with W^t and V left and right eigenvectors of DF and z a component of (s, w, λ). Thus we need only compute a few of the second derivatives of F to obtain H_z. These have been computed with finite differences in our codes thus far. (ADOL-C cannot be called recursively, so functions whose definition invokes ADOL-C cannot be differentiated using ADOL-C.)

Once we've computed the necessary second derivatives, we assemble the Jacobian of the system, and Newton's method is straightforward. As before, the convergence of Newton's method appears to be limited by the condition number of K. Although DF becomes singular as we approach the bifurcation point, the condition number of K is generally well-behaved. The mesh is adapted periodically during these computations as follows. To add mesh points, we begin by computing the L_∞ error $\max |p'(s_j) - f(p(s_j))|$ for a set of evenly spaced points $\{r_j\} \in [-s_i, s_i]$ in each mesh interval. If, for a particular interval joining the i-th and $(i+1)$-st mesh points, the error is greater than a specified multiple of the average error for all intervals, we add a new mesh point at the average of the points obtained by shooting forward from the i-th mesh point and shooting backward from the $(i + 1)$-st meshpoint. To remove mesh points, we check for each mesh point the resulting error if that point is removed, and remove it if the new error is no more than a prescribed tolerance relative to the sum of the previous errors.

The length of this code is about 600 lines of MATLAB (which includes the version of the algorithm to converge to a simple periodic orbit), and three C++ files, for evaluation of the vector field and the interface between MATLAB and ADOL-C. The Appendix gives a more complete description of this algorithm in terms of pseudocode that corresponds closely to our MATLAB implementation.

Figure 2: Three different methods have been used to compute a periodic orbit that lies in a polynomial curve in the plane. Distance from the periodic orbit is plotted as a function of time during one traversal of the periodic orbit. The numerical periodic orbit in the top panel was computed with AUTO, the three in the middle panel with a fourth order Runge-Kutta method, and the bottom one with a multiple shooting algorithm employing automatic differentiation.

5 Case Studies

In this section, we present comparisons of computations on several test problems chosen to evaluate different aspects of the algorithms.

5.1 A periodic orbit in an algebraic curve

The first test problem was chosen so that the location of its periodic orbit is explicitly given by an algebraic formula. The vector field is defined as

$$\begin{aligned} \dot{x} &= y - y^2 - x(x^2 - y^2 + 2y^3/3 + c) \\ \dot{y} &= x + (y - y^2)(x^2 - y^2 + 2y^3/3 + c) \end{aligned}$$

This vector field has a stable periodic orbit that lies in the zero set of the polynomial $g(x, y) = x^2 - y^2 + 2y^3/3 + c$ when $c \in (0, 1/3)$. Thus, evaluation of g along computed trajectories is an explicit measure of their accuracy. We shall call $\max |g(u(t)|$ the **residual** of a numerically computed orbit $u(t)$, where the residual map is evaluated as a discrete function at mesh points or regarded as a continuous function using a mapping of discrete closed curves into a function space.

For $c = 0.07$, we compared three ways of computing the periodic orbit: the collocation method implemented in the AUTO package, numerical integration with a fourth order Runge-Kutta algorithm, and a multiple shooting code that uses automatic differentiation.

The computer program AUTO [5] is widely regarded as the best available tool for the computation of periodic orbits. AUTO uses a collocation method with between 2 and 7 collocation points per mesh interval to compute orbits. The algorithm used by AUTO is superconvergent at the mesh points. The top panel of Figure 2 displays the value of the polynomial h along a periodic orbit computed by AUTO using 60 mesh intervals and 4 collocation points per interval. The maximum value of $|h|$ at the mesh points is approximately 6×10^{-11}. We have been unable to achieve significantly greater accuracy with AUTO in varying the number of mesh intervals (up to 200) or the number of internal collocation points (up to 7) in this example.

The "standard" fourth order Runge-Kutta algorithm [16] for the system $\dot{x} = f(x)$ with step size h is described by the following formulas that yield the approximation x_1 to $\phi_h(x_0)$:

$$\begin{aligned} k_1 &= hf(x_0) \\ k_2 &= hf(x_0 + k_1/2) \\ k_3 &= hf(x_0 + k_2/2) \\ k_4 &= hf(x_0 + k_3) \\ x_1 &= x_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

Setting $x_1 = \rho_h(x_0)$, $\rho_h : R^n \to R^n$ is a map that approximates the flow map ϕ_h to fourth order in the step size h. This implies that the errors made in computing one step of the algorithm are comparable to h^5 and that the errors associated with computing a trajectory for bounded time are comparable to h^4. When h is very small, round off errors dominate the truncation errors in the finite difference calculations of derivatives of f that are implicit in the algorithm. As h is reduced, we observed that there is an optimal stepsize h_o for accuracy at which the round off and truncation errors appear to be balanced. This is illustrated by data from numerical experiments in which the Runge-Kutta algorithm is used to evolve the trajectory with initial point $(0, 0.2952161257895192)$ with different step sizes. The initial point lies on the periodic orbit to within round-off error, and the trajectory was computed until it returned to the y-axis near the initial point.

The apparent value of h_o is approximately 0.001. For stepsizes larger than 0.001, the residual of the computed orbits decrease with h at a rate that appears consistent with the fourth order convergence of the Runge-Kutta algorithm. For step sizes smaller than 0.001, the residual increases in an erratic manner. The second panel of Figure 2 shows the value of g as a function of time along Runge-Kutta trajectories with step sizes 0.00125, 0.001 and 0.0001. The smallest observed value of the residual was approximately 8×10^{-15} with step size 0.001. Since the orbit has period approximately 7.7, the numerical integration with step length 0.001 required over 7700 steps and approximately $30,000$ function evaluations.

The bottom panel of the figure shows the results of a calculation using a symmetric multiple shooting algorithm employing automatic differentiation. There are five mesh intervals and the maximum value of $|h|$ is approximately 6×10^{-16}. The degree of the Taylor series polynomials used in the calculation is 16. Convergence is obtained in approximately 6 Newton steps starting with mesh points that are far from the computed solution. Thus our method produces solutions to this problem that are an order of magnitude more accurate than those produced by the most accurate numerical integration possible with the standard fourth order Runge-Kutta algorithm and five orders of magnitude more accurate than those produced with AUTO, a widely used collocation algorithm. Given the differences in the way in which each of these algorithms was implemented, direct comparisons of their efficiency were not made. Nonetheless, it is apparent that the small mesh sizes and rapid convergence of the Taylor series method make it very efficient compared to other methods of computing periodic orbits to high accuracy.

Figure 3: Four nested periodic orbits in the vector field $y\partial_x + ((0.897258546 - x^2)y - (x^3 + 0.87x^2 + -1.127921667x - 1))\partial_y$ The three inner orbits are not resolved in the large figure. The inset shows a small region near the left intersection of the inner orbits with the x-axis.

5.2 Bifurcations in a planar vector field with multiple limit cycles

Dangelmayr and Guckenheimer [3] investigated the dynamics of the four parameter family of vector fields:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= -(x^3 + rx^2 + nx + m) + (b - x^2)y \end{aligned}$$

The system describes the effect of symmetry imperfections in the unfolding of a codimension two bifurcation of an equilibrium with symmetry rotation by π. The dynamics of this family is surprisingly complex, with over twenty different inequivalent phase portraits. In a small region of the parameter space, there are phase portraits with a single equilibrium point and four nested limit cycles. The existence of this parameter region was deduced from an analysis of unfoldings of subsidiary codimension three bifurcations in Dangelmayr and Guckenheimer [3]. Parameter values at which this phase portrait occurs were first determined by Malo [18]. Malo developed simple shooting algorithms to trace curves of saddle-node bifurcations of periodic orbits. Applying these to the above system, he demonstrated that, when $r = 0.87$ and $m = -1$ the parameter region in the (n, b) plane for which the system has four nested limit cycles is a strip of width approximately $3 * 10^{-9}$. We used this example to test our algorithms for computation and continuation of saddle-node bifurcations of periodic orbits.

Figure 3 shows periodic orbits of the vector field for parameters $n = -1.127921667$ and $b = 0.897258546$. The inset shows details of the three inner periodic orbits near their left intersection with the x axis. The saddle-node bifurcation points in this system were computed with the symmetric shooting method described earlier. However, due to the sensitivity of the problem, it was a rather difficult undertaking. The shape of the three inner curves makes the computation difficult, as there is a very slow, tight turn, and a very fast, wide turn in the orbits. In order to converge from a mesh of regularly spaced points, we needed to decimate the mesh several times before applying Newton steps. In this way, we were able to obtain representations for each of the nested periodic orbits. Each of these representations uses about 20 to 30 mesh points. Each periodic orbit of the system in the vicinity of these parameter values intersects the x axis vertically in two points. Moreover, for each value of x, there is a unique value of the parameter b for which the trajectory intersects the horizontal axis at x.

Only the middle of the three inner orbits yielded starting data sufficient to converge directly to a saddle-node bifurcation. Since we could only find one of the saddle-node bifurcations, we used a continuation strategy to find the other. We continued the first saddle-node

n	b_1	b_2	$b_2 - b_1$
-1.127921667	8.9725854413042e-01	8.9725854695863e-01	2.82821 e-09
-1.127921669	8.9725854512245e-01	8.9725854795178e-01	2.82933 e-09
-1.127921671	8.9725854611447e-01	8.9725854894492e-01	2.83045 e-09
-1.127921673	8.9725854710650e-01	8.9725854993807e-01	2.83157 e-09
-1.127921675	8.9725854809852e-01	8.9725855093121e-01	2.83269 e-09
-1.127921677	8.9725854909055e-01	8.9725855192436e-01	2.83381 e-09
-1.127921679	8.9725855008257e-01	8.9725855291751e-01	2.83494 e-09
-1.127921681	8.9725855107459e-01	8.9725855391065e-01	2.83606 e-09
-1.127921683	8.9725855206662e-01	8.9725855490380e-01	2.83718 e-09
-1.127921685	8.9725855305864e-01	8.9725855589694e-01	2.83830 e-09

Table 1: Computed values of saddle-node bifurcation of the cubic vector field. The first column gives values of the parameter n, the second and third columns are computed values of b at the two curves of saddle-node bifurcation for the specified value of n, and the fourth column is the difference between the two values of saddle-node bifurcation.

curve with increasing n until we reached the neighborhood of an apparent cusp point. We then hoped to jump onto the second saddle-node curve and continue back to the original value of n. The computation to the cusp was quite easy: we were able to take large (size 2×10^{-6}) steps in n at first, and were able to quickly come close to the cusp point, gradually decreasing our stepsizes to 10^{-8}. Once we were close to this point, we took a step of size -2.7×10^{-8}, from where the routines converged to a saddle-node bifurcation on the new curve. Following this curve back to the original value of n was significantly more difficult than computing the previous curve. We were unable to take large stepsizes when we were close to the cusp because large steps converged back to the previously computed saddle-node curve. The computation was also quite sensitive to initial conditions, so we often couldn't take uniform steps. The Jacobians of the augmented systems along this curve are generally on the order of 10^{10}. Consequently, our approximations to these orbits generally have an L_∞ error on the order of 10^{-5}. Values we have computed for a portion of the saddle-node curves near Malo's parameters are given in Table 1.

5.3 Multiple time scales and canards

The term canard describes trajectories of singularly perturbed systems that have segments which are close to unstable slow manifolds. Inspiration for the term comes from limit cycles of the dynamical system

$$\begin{aligned} \dot{x} &= (y - x^2 - x^3)/\epsilon \\ \dot{y} &= a - x \end{aligned}$$

As $\epsilon \to 0$, this system has a slow manifold that approaches the cubic curve $y = x^2 + x^3$. The fast vector field is horizontal in the limit $\epsilon = 0$. The portion of the slow manifold between $x = -2/3$ and $x = 0$ is unstable, while the remainder is stable. When $a = 0$, there is an equilibrium point at the origin where the fast vector field is tangent to the slow manifold. Hopf bifurcation occurs at this point. As a decreases from 0, a family of stable periodic orbits grows rapidly with the magnitude of a. The orbits quickly stabilize as relaxation oscillations approximated by stable segments of the slow manifold and horizontal segments connecting the origin with $(-1, 0)$ and (-2/3,4/27) with (1/3,4/27). The periodic orbits of intermediate amplitude take the shape of canards in which there is a segment that follows the unstable portion of the slow manifold. An extensive analysis of the canard solutions has been undertaken from the three different perspectives of nonstandard analysis [4], classical asymptotic analysis [7] and geometric singular perturbation theory [6]. Here we investigate the numerical computation of canards with our adaptive step length multiple shooting algorithm and compare these results with AUTO calculations.

The canard phenomenon is subtle. We recall a few of the results from the theory. Fenichel [8] established the existence of a slow manifold for positive values of ϵ as well as $\epsilon = 0$. Eckhaus [7] computed asymptotic expansions for the slow manifold. Its distance from the cubic curve $y = x^2 + x^3$ has order ϵ. Away from the turning points, trajectories flow towards or away from the cubic characteristic at exponential rates of order $1/\epsilon$. A very large region of initial points has trajectories that follow the unstable branch of the slow manifold for approximately the same distance and then jump back to a stable branch at approximately the same height. Asymptotic formulas for this critical height have been obtained. The critical height is extremely sensitive to the value of the parameter a, varying at a rate comparable to $\exp(-Q/\epsilon)$ where Q is given by the asymptotic theory. The critical height determines the size of a canard. All trajectories starting near the stable left branch of the slow manifold flow to the local maximum of the cubic characteristic and then jump to the right branch of the slow manifold. They then follow the right branch to the local minimum of the cubic curve. If the parameter a is in the range in which the canards "with heads" occur, the trajectory then climbs the unstable portion of the slow manifold to the critical height before jumping back to the left branch of the slow manifold. Thus, all trajectories that begin on the left branch of the slow manifold are swept into a small neighborhood of the canard cycle before they have completed a single circuit around the cycle. Nonetheless, tiny variations in an initial condition near the minimum of the cubic characteristic produce trajectories that separate from each other after traveling only a short distance along the unstable branch of the slow manifold.

Instability of the middle branch of the slow manifold prevents computation of canards with numerical integration. To make our discussion concrete, consider what happens with $\epsilon = 0.001$. The Jacobian of the vector field is

$$\begin{pmatrix} 1000(-2x - 3x^2) & 1000 \\ -1 & 0 \end{pmatrix}$$

When $x \in [-1/2, -1/6]$ and $a < 0$, the x components of nearby trajectories separate at a rate at least $\exp(250t)$ while the y component of the vector field increases at a rate at

Figure 4: Fourth order Runge-Kutta integration of the canard vector field. Due to the massive instability of the unstable branch of the slow manifold, the integration is unable to compute trajectories that follow this branch. Compare with the family of canards shown in Figure 5.

Figure 5: A family of canards computed with the forward multiple shooting algorithm described in Section 3. The parameter $\epsilon = 0.001$.

most 1/2. Thus a trajectory requires at least time 2/3 to traverse the portion of the slow manifold with $x \in [-1/2, -1/6]$. During this time, the relative separation of trajectories in the x direction increases by a factor of well over $\exp(150)$. Thus, if an initial condition has distance greater than $\exp(-150)$ from the slow manifold, it will not track it until x reaches $-1/2$. It is evident that extended precision beyond the 53 bit precision of IEEE-754 floating point arithmetic is required for this task. If we try to follow the unstable portion of the slow manifold by choosing initial conditions that lie as close together as possible on opposite sides of the manifold, then Runge-Kutta integration was observed to return to opposite sides of the unstable manifold erratically. Figure 4 shows such a "chaotic" trajectory. The numerical integration algorithm has qualitative behavior inconsistent with the trajectories of any planar vector field. There is no numerical trajectory that approximates the canards even crudely.

We used a forward multiple shooting method and the program AUTO to compute this family of canards. The AUTO parameters were set to use two hundred mesh intervals and error tolerances of 10^{-12}. Starting at the Hopf bifurcation point, AUTO was able to compute the entire family of canard orbits. Figure 5 shows selected orbits from this family. We sought to evaluate the precision of the AUTO computations. To achieve this goal we took the trajectory of one canard computed in AUTO as an initial discrete closed curve for computation with an adaptive step length forward multiple shooting algorithm. This algorithm first computes Taylor polynomial approximations to the trajectory segments of the discrete closed curve obtained from AUTO. These computations yield an estimate of approximately 5×10^{-9} for the maximum mismatch from the end of one trajectory segment to the initial point of the next, despite the stringent error tolerances used in the calculation and the superconvergence at mesh points of the AUTO collocation algorithm. The condition number of the Newton method Jacobian at this orbit was approximately 1×10^{50} with the parameter a fixed. The algorithm did not converge. However, fixing the period and varying a, the condition number was approximately 3.5×10^{7} and one Newton step produced an approximate discrete closed curve with the distance from the endpoint of one trajectory segment to the initial point of the next having magnitude smaller than 6×10^{-15}. The maximum length of the difference between the tangent vector to the Taylor polynomials and the vector field at the ends of the time steps is approximately 1×10^{-13}. Dividing the length of the difference by the length of the vector field to produces an estimate of approximately 2×10^{-13} for the maximum deviation of the approximate periodic orbit from the flow direction.

Figure 6: This three dimensional plot gives the difference between a canard computed with AUTO and a canard computed with a Taylor series, forward shooting method. The canard is plotted in the horizontal plane and the magnitude of the difference between the two computations is plotted on the vertical axis.

The distance between the discrete closed curve computed by AUTO and the discrete closed curve produced by the multiple shooting algorithm is approximately 1.4×10^{-7}. Figure 6 displays a three dimensional plot of the difference of the y coordinates of the two discrete trajectories. The differences between the x coordinates of the two trajectories are much smaller. The largest differences occur when the trajectory leaves the slow manifold and appear smooth enough that it is unlikely that the differences are due to round-off error. AUTO calculations with coarser meshes yield a similar pattern, with larger differences that have a similar shape along the jumps of the canard. Thus, we believe that our calculation is sufficiently precise that the difference between the two trajectories is an estimate for the distance of the mesh points computed by AUTO from the actual periodic orbit. The parameter values imported from AUTO for the canards had eleven significant digits. The parameter value obtained with Newton's method differs from the AUTO parameter by approximately 1.5×10^{-15}, agreeing to the last digit of the value from AUTO. We next used the period as a continuation parameter in order to track the family of canards. These calculations gave parameter values that differ in the thirteenth significant digit, and they do so erratically. Thus, the sensitivity of the canards to parameter variations is so great that our algorithms are unable to compute the qualitative behavior of this parameter variation. This is hardly surprising since the asymptotic theory described above estimates of the parameter variation across the canard family to be smaller than 10^{-20}. Calculation of the parameter variation appears to be beyond the bounds that are readily feasible without using greater precision than IEEE-754 double precision floating point arithmetic. Our attempts to construct continuation codes for tracking the canard solutions have not achieved the robustness we seek. However, the orbits themselves appear to be computed with precision that is dominated by round-off errors as in the case of non-stiff systems.

5.4 Monodromy in a model of coupled Josephson junctions

Swift and Watanabe [23] studied the dynamics of arrays of N Josephson junction oscillators shunted by a series LRC load, a system symmetric with respect to all interchanges of the oscillators. These systems have "splay-phase" solutions in which the N oscillators all undergo the same motion but out of phase with one another. Denoting the motion of oscillator i in the splay phase state by $\phi_i(t)$, the oscillators can be numbered so that $\phi_{i+1}(t) = \phi_i(t+2\pi/N)$. In the limit that the Stewart-McCumber parameter of these systems is zero, the splay phase state is neutrally stable. Numerical integration of the system with other values of this parameter suggested that the splay phase states might be neutrally stable for positive values of the Stewart-McCumber parameter as well, with a return map possibly having eigenvalue 1 with

multiplicity $N-3$. Swift and Watanabe studied the case of four coupled junctions carefully and demonstrated that the splay phase states are represented by hyperbolic periodic orbits for generic values of the Stewart-McCumber parameter. They did so by computing the splay phase states using AUTO86 and then computing the monodromy of the periodic orbit outside of AUTO. (The computation of monodromy matrices has been greatly improved in AUTO97 compared to AUTO86, but AUTO97 was not available when Swift and Watanabe did their work.)

The vector field f that represents an array of four junctions is defined by the ten dimensional system:

$$\begin{aligned}
\dot{x}_1 &= x_5 \\
\dot{x}_2 &= x_6 \\
\dot{x}_3 &= x_7 \\
\dot{x}_4 &= x_8 \\
\dot{x}_5 &= (I - x_5 - sin(x_1 - x_{10}))/b \\
\dot{x}_6 &= (I - x_6 - sin(x_2 - x_{10}))/b \\
\dot{x}_7 &= (I - x_7 - sin(x_3 - x_{10}))/b \\
\dot{x}_8 &= (I - x_8 - sin(x_4 - x_{10}))/b \\
\dot{x}_9 &= x_{10} \\
\dot{x}_{10} &= ((x_1 + x_2 + x_3 + x_4)/4 - rx_{10} - x_9/c)/l
\end{aligned}$$

Since Swift and Watanabe had difficulty computing the splay phase states and their stability, we sought to confirm their results with an automatic differentiation based solver. We used the Hermite interpolation global method to compute the splay phase state, starting with linear varying phase for each oscillator. The parameter values were $I = 2.5, l = 0.75, r = 0, c = 20$ and Stewart-McCumber parameter $b = 0.2$. With twenty mesh intervals and degree sixteen Taylor series approximations at the mesh points, the algorithm converged in six Newton steps to a an approximate periodic orbit $p(t)$. Using the uniform norm in R^{10}, the norm of $p'(t) - f(p(t))$ appears to be smaller than 2×10^{-13}. The eigenvalues of the monodromy matrix were computed to be

$$\begin{aligned}
&1.212564610112479e-06 \pm 5.700237500982539e-08i \\
&1.390021921820548e-06 \\
&-1.172334117548194e-03 \pm 4.455213497385255e-04i \\
&8.826221531499485e-01 \\
&1.000000000000006e+00 \\
&1.003009060195232e+00 \\
&1.149723251975266e+00 \pm 5.356810539765165e-02i
\end{aligned}$$

The monodromy matrix of a periodic orbit has an eigenvalue 1. The precision with which this eigenvalue is computed is a measure for the accuracy of the computation of the periodic orbit and its monodromy matrix. In our calculation, the error for this eigenvalue is $6 \times$

10^{-15}. We also note that the eigenvalue 1.003009060195232 is in excellent agreement with the perturbation analysis of Swift and Watanabe [23], Figure 8. As a final check on the precision of these calculations, we repeated them with a coarser mesh having ten mesh intervals. The algorithm converged in seven Newton steps. The monodromy matrix agreed with the values in the table above to ten decimal places. The error in the eigenvalue 1 was approximately 5.5×10^{11}. Thus, these periodic orbit calculations seem to be converging quickly and accurately.

6 Appendix

List of Algorithms

References

[1] U. Ascher, R. Mattheij and R. Russell, Numerical Solution of Boundary Value Problems, Prentice Hall, 1988.

[2] I. Babushka,

Algorithm 1 Continue Saddle Node Bifurcations

Load data and parameters
Set global variables
Call procedure Find Orbit
for number of continuation points requested **do**
  Call procedure Find SNB
  Call procedure Symmetric Compute Return Map Derivative
  Save representation for located periodic orbit: Taylor coeffs, total derivatives, $data, delta, params$, and $ReturnMapDerivative$
  Add continuation stepsize to continuation parameter
  if we have another point to find **then**
    Call procedure Symmetric Predict Next SNB

Procedure 2 Find Orbit

for 1 to *steps* **do**
  Call procedure Newton Init
  Call procedure Symmetric Newton Loop
  Call procedure Symmetric Decimate
  Call procedure Newton Init
  Call procedure Symmetric Newton Loop
  for 1 to $rsteps$ **do**
    Call procedure Symmetric Refine
    Call procedure Newton Init
    Call procedure Symmetric Newton Loop
    Let $memax$ be the max L_∞ error over all intervals
    if $memax$ is small enough **then**
      quit Find Orbit

Procedure 3 Newton Init

Clear matrix collections $E, D, Q, Tay, JMat, FullJMat$, and column matrix err
for all mesh points **do**
  Compute Taylor series coefficients and total derivatives for the trajectories through the mesh point using ADOL-C
  Compute an orthonormal basis for a cross-section at the mesh point by computing an orthogonal matrix Q at each mesh point with the first vector parallel to the vector field

Procedure 4 Symmetric Newton Loop

for 1 to $nsteps$ **do**
  Call procedure Symmetric Evaluate
  Call procedure Symmetric Newton Step

Procedure 5 Symmetric Evaluate

Clear matrix collections E, D, and fd
Clear matrix err
for all mesh points **do**
  Compute Taylor series coefficients and total derivatives for the trajectories through the mesh point using ADOL-C
for all intervals i **do**
  Let err(column i) be the "shooting error": the difference between shooting forward from the i-th mesh point and shooting backward from the $(i+1)$-st mesh point
  Compute the Jacobians of the shooting error WRT the i-th and $(i+1)$-st mesh points
  Compute the Jacobian of the shooting error WRT the interval stepsize
  Form the Jacobians of the shooting error WRT the bases of cross-section variables and the interval stepsizes (matrices $D\{i\}$ and $E\{i\}$)
  Let $fd\{i\}$ be the Jacobian of the approximate flow map from mesh point i to mesh point $i+1$: multiply the R^n Jacobian of the flow map forward from the i-th mesh point by the inverse of the R^n Jacobian of the flow map backward from the $(i+1)$-st mesh point

Procedure 6 Symmetric Newton Step

Stack the columns of err into vector errv
Form the Jacobian J from the matrix collections D and E, with the matrices $D\{i\}$ on the diagonal and the matrices $E\{i\}$ on the superdiagonal
Solve $J * Nstep = errv$ for the Newton update $Nstep$ (in cross-section coordinates)
Transform the Newton update back to R^n coordinates and update *data* and *delta*

Procedure 7 Symmetric Decimate

Call procedure Symmetric Interpolate Full
Set $adddata$ to be a $2nint \times dim + 1$ array of zeros
Set $j = 1$, $id = 1$
while $id \leq nint$ **do**
  $adddata(\text{row}j) = data(\text{row}id)$
  if $id < nint$ **then**
    if adjacent L_∞ errors are small enough **then**
      Calculate the L_∞ error if mesh point i is removed
      if the error is small enough **then**
        increment id, effectively removing mesh point i
  Increment id and j
Save the last endpoint
Replace *data* by $adddata$ and reset *delta*

Procedure 8 Symmetric Interpolate Full

Clear necessary arrays
for all intervals i **do**
  Let $ppts$ be an evenly spaced set of times in $[-delta(i), delta(i)]$
  Let $xpts$ be the values of the shooting polynomial for the times of $ppts$
  Let $xdpts$ be the values of the derivative of the shooting polynomial for the points of $ppts$
  Let $vpts$ be the values of the vector field at the points of $xpts$
  Let $errpts$ be the differences between $vpts$ and $xdpts$
  Let $emax(i)$ be the largest absolute value in $errpts$
Let $eavg$ be the average L_∞ error over all of the intervals

Procedure 9 Symmetric Refine

Call procedure Symmetric Interpolate Full
Set $adddata$ to be a $2nint \times dim + 1$ array of zeros
Set $j = 1$
for $ir = 1$ to $nint$ **do**
  $adddata(\text{row}j) = data(\text{row}ir)$
  if $(emax(ir) > (errortolerance) * eavg)$ **then**
    Add a mesh point at the average of the forward and backward trajectories from the endpoints
  Increment j
Save the last endpoint
Replace $data$ by $adddata$ and reset $delta$

Procedure 10 Find Saddle Node Bifurcation

for 1 to maxsnbsteps **do**
  Call procedure Newton Init
  Call procedure Symmetric SNB Loop
  Call procedure Symmetric Decimate
  Call procedure Newton Init
  Call procedure Symmetric SNB Loop
  for 1 to rsteps **do**
    Call procedure Symmetric Refine
    Call procedure Newton Init
    Call procedure Symmetric SNB Loop
    Let $memax$ be the largest L_∞ error over all intervals
    if $ReturnMapDerivative$ is close enough to 1 and $memax$ is small enough **then**
      quit Find Saddle Node Bifurcation

Procedure 11 Symmetric Newton SNB Loop

for 1 to $nsteps$ **do**
  Call procedure Symmetric Newton SNB Init
  Call procedure Create Bordered Matrix
  Call procedure Symmetric Newton SNB Step

Procedure 12 Symmetric Newton SNB Init

Clear $E, D, Q, Tay, JMat, FullJMat, Err, Fullderrm, Fullderrp, fd$
for all mesh points **do**
  Compute Taylor series coefficients and total derivatives for the trajectories through the mesh point using ADOL-C
  Compute an orthonormal basis for a cross-section at the mesh point by computing an orthogonal matrix Q at each mesh point with the first vector parallel to the vector field
for all intervals i **do**
  Let err(column i) be the "shooting error": the difference between shooting forward from the i-th mesh point and shooting backward from the $(i+1)$-st mesh point
  Compute the Jacobians of the shooting error WRT the i-th and $(i+1)$-st mesh points
  Compute the Jacobian of the shooting error WRT the interval stepsize
  Form the Jacobians of the shooting error WRT the bases of cross-section variables and the interval stepsizes (matrices $D\{i\}$ and $E\{i\}$)
  Let $fd\{i\}$ be the Jacobian of the approximate flow map from mesh point i to mesh point $i+1$: multiply the R^n Jacobian of the flow map forward from the i-th mesh point by the inverse of the R^n Jacobian of the flow map backward from the $(i+1)$-st mesh point

Procedure 13 Create Bordered Matrix

Create matrix M of size $nint * dim + 1 \times nint * dim + 1$
Put J in the upper left $(nint * dim \times nint * dim)$ corner of M
Put a 1 in the upper right and lower left corners of M
if M is singular **then**
  exit the program {something is seriously wrong}
Solve system $MV = [00...01]^T$ for V
Set $G = V(nint * dim + 1, 1)$
Solve system $\widetilde{W}M = [00...01]$ for $\widetilde{W}$
Set $W = \widetilde{W}$(columns 1 to $nint * dim$)
  as a check, $G = \widetilde{W}(1, nint * dim + 1)$

Procedure 14 Symmetric Newton SNB Step

Set $Kbar$ to be an $(nint * dim + 1) \times (nint * dim + 1)$ matrix of zeros
Put matrix M in the upper left $(nint * dim \times nint * dim)$ block of $Kbar$
Stack the columns of err into vector $errv$
for all mesh points **do**
  Compute $\frac{\partial(JMat)}{\partial(freeparam)}$ using finite differencing, and store the results
for all intervals i **do**
  Compute the Hessians of the shooting error WRT the free parameter and the i-th and $(i+1)$-st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(freeparam)}$
  for all cross-section variables **do**
    Compute the Hessian of the shooting error WRT the cross-section variable and the i-th and $(i+1)$-st mesh points (in cross-section coordinates)
  Compute the Hessian of the shooting error WRT $delta(i)$ and the i-th and $(i+1)$-st mesh points (in cross-section coordinates)
  Compute the Hessian of the shooting error WRT $delta(i)$
  Compute the Hessian of the shooting error WRT $delta(i)$ and the free parameter
Set $bigKz$ to be an $nint * dim \times nint * dim$ matrix of zeros
Fill $bigKz$ with the Hessians that involve the free parameter and either a mesh point or $delta(i)$ for some i
Set $Kbar(nint * dim + 1, nint * dim + 1) = -W * bigKz * V$(rows 1 to $nint * dim$)
for all mesh points i **do**
  for all cross-section variables k at mesh point i **do**
    Fill Kz with the Hessians that involve cross-section variable k at mesh point i
    Set $Kbar(nint * dim + 1, (i-1) * dim + k) = -W * Kz * V$
  Fill Kz with the Hessians that involve $delta(i)$
  Set $Kbar(nint * dim + 1, (i-1) * dim) = -W * Kz * V$
for all intervals i **do**
  Compute the Jacobian of the shooting error WRT the free parameter
  Insert this vector into the appropriate rows of the last column of $Kbar$
Solve $Kbar * Nstep = [errv; G]$ for Newton update Nstep
Transform Newton update to R^n coordinates and update $data$ and $delta$
Update the free parameter by subtracting $Nstep(nint * dim + 1)$ from it

Procedure 15 Symmetric Compute Return Map Derivative

Set $Monodromy$ to be a $dim \times dim$ identity matrix
for all intervals i **do**
  Multiply $Monodromy$ on the left by $fd\{i\}$ (which was computed in Symmetric Evaluate)
Transform $Monodromy$ to $ReturnMapDerivative$ by multiplying on the left and right by the matrices to transform to cross-section coordinates

Procedure 16 Symmetric Predict Next SNB

Call procedure Symmetric Newton SNB Init
Call procedure Create Bordered Matrix
Set $Kbar$ to be an $nint * dim + 1 \times nint * dim + 1$ matrix of zeros
Put matrix M in the upper left $(nint * dim \times nint * dim)$ block of $Kbar$
Stack the columns of err into vector $errv$
for all mesh points **do**
  Compute $\frac{\partial(JMat)}{\partial(freeparam)}$ using finite differencing, and store the results
  Compute $\frac{\partial(JMat)}{\partial(contparam)}$ using finite differencing, and store the results
for all intervals i **do**
  Compute the Hessians of the shooting error WRT the free parameter and the i-th and $(i+1)$-st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(freeparam)}$
  Compute the Hessians of the shooting error WRT the cont parameter and the i-th and $(i+1)$-st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(contparam)}$
  for all cross-section variables **do**
    Compute the Hessian of the shooting error WRT the cross-section variable and the i-th and $(i+1)$-st mesh points (in cross-section coordinates)
  Compute the Hessian of the shooting error WRT $delta(i)$ and the i-th and $(i+1)$-st mesh points (in cross-section coordinates)
  Compute the Hessian of the shooting error WRT $delta(i)$
  Compute the Hessian of the shooting error WRT $delta(i)$ and the free parameter
  Compute the Hessian of the shooting error WRT $delta(i)$ and the cont parameter
Set $bigKz$ to be an $nint * dim \times nint * dim$ matrix of zeros
Fill $bigKz$ with the Hessians that involve the free parameter and either a mesh point or a $delta(i)$ for some i
Set $Kbar(nint * dim + 1, nint * dim + 1) = -W * bigKz * V$(rows 1 to $nint * dim$)
Set $bigKz$ to be an $nint * dim \times nint * dim$ matrix of zeros
Fill $bigKz$ with the Hessians that involve the cont parameter and either a mesh point or a $delta(i)$ for some i
Set $errv(nint * dim + 1) = -W * bigKz * V$(rows 1 to $nint * dim$)
for all mesh points i **do**
  for all cross-section variables k at mesh point i **do**
    Fill Kz with the Hessians that involve cross-section variable k at mesh point i
    Set $Kbar(nint * dim + 1, (i-1) * dim + k) = -W * Kz * V$
  Fill Kz with the Hessians that involve $delta(i)$
  Set $Kbar(nint * dim + 1, (i-1) * dim) = -W * Kz * V$
for all intervals i **do**
  Compute the Jacobian of the shooting error WRT the free parameter
  Insert this vector into the appropriate rows of the last column of $Kbar$
  Compute the Jacobian of the shooting error WRT the cont parameter
  Insert this vector into the appropriate rows of the vector $errv$
Solve $Kbar * Nstep = [errv; G]$ for Newton update Nstep
Transform Newton update to R^n coordinates and update $data$ and $delta$
Update the free parameter by subtracting $Nstep(nint * dim + 1)$ from it

[3] G. Dangelmayr and J. Guckenheimer, On a four parameter family of planar vector fields, Arch. Rat. Mech. Anal., 97, 321-352, 1987.

[4] M. Diener, Canards et bifurcations, in *Mathematical tools and models for control, systems analysis and signal processing, Vol. 3 (Toulouse/Paris, 1981/1982)*, 289–313, CNRS, Paris.

[5] E. Doedel, AUTO, ftp://ftp.cs.concordia.ca/pub/doedel/auto.

[6] F. Dumortier and R. Roussarie, Canard cycles and center manifolds, Mem. Amer. Math. Soc. **121** (1996), no. 577, x+100 pp.

[7] W. Eckhaus, A standard chase on French ducks, Lect. Notes in Math. 985, 449-494, 1983.

[8] N. Fenichel, Persistence and smoothness of invariant manifolds for flows, Indiana Univ. Math. J. 21, 193-226, 1971

[9] W. Govaerts and J.D. Pryce, A singular value inequality for block matrices, Lin. Alg. Appl. 125, 141 - 148, 1989.

[10] W. Govaerts, J. Guckenheimer and A. Khibnik, Defining functions for multiple Hopf bifurcations, SIAM J. Num. Anal., SIAM J. Num. Anal., 34, 1269-1288, 1997.

[11] Govaerts, W. and Pryce, J.D. (1989), A singular value inequality for block matrices, Lin. Alg. Appl. 125, 141 - 148.

[12] Griewank, A., Juedes, and J. Utke, 'ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++', Version 1.7, Argonne National Laboratory.

[13] J. Guckenheimer and W. G. Choe, Computing periodic orbits with high accuracy, Computer Methods in Applied Mechanics and Engineering, 170, 331-341, 1999.

[14] J. Guckenheimer and P. Holmes, Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields, Springer-Verlag, 1983.

[15] J. Guckenheimer, M. Myers and B. Sturmfels, Computing Hopf Bifurcations I, SIAM J. Num. Anal., 34, 1-21, 1997.

[16] E. Hairer and G. Wanner, Solving Ordinary Differential Equations I, 2nd. ed., Springer-Verlag, 1993.

[17] V. Kuznetsov and V. Levitin, CONTENT, http://www.cwi.nl

[18] S. Malo, Rigorous Computer Verification of Planar Vector Field Structure, Thesis, Cornell University, 1993.

[19] Matlab is a product of The MathWorks, Inc.

[20] Pitcon, ...

[21] J. Shen and G. Strang, Asymptotic analysis of Daubechies polynomials, Proc. Am. Math. Soc., 124, 3819-3833, 1996.

[22] J. Shen and G. Strang, personal communication.

[23] S. Watanabe and J. Swift, Stability of periodic solutions in series arrays of Josephson junctions with internal capacitance, J. Nonlinear Sci. 7, 503-536, 1997.