Physica D 171 (2002) 127–137

PHYSICA D

www.elsevier.com/locate/physd

Computing accurate Poincaré maps

Warwick Tucker[*]

Department of Mathematics, Uppsala University, Box 480, 751 06 Uppsala, Sweden

Received 27 August 2001; received in revised form 9 July 2002; accepted 12 July 2002
Communicated by E. Ott

Abstract

We present a numerical method particularly suited for computing Poincaré maps for systems of ordinary differential equations. The method is a generalization of a stopping procedure described by Hénon [Physica D 5 (1982) 412], and it applies to a wide family of systems.

Keywords: Poincaré maps; Numerical method; Differential equations

1. Introduction

In this paper, we will consider an autonomous system of differential equations

$$\dot{x} = f(x), \quad x \in \mathbb{R}^n. \tag{1}$$

Throughout the text, we will assume that the vector field f is sufficiently smooth (C^1 will do nicely). Using standard notation, let $\varphi(x, t)$ denote the solution of (1) satisfying $\varphi(x, 0) = x$. The curve $\mathrm{O}(x) = \{\varphi(x, t) : t \in \mathbb{R}\}$ is called the *orbit* or *trajectory* passing through the point x.

Traditionally, when an orbit of (1) is to be approximated via a numerical method, it is done in small time increments. Given an initial value $(x^{(0)}, t^{(0)})$, the numerical approximation yields successive grid points $(x^{(k)}, t^{(k)})$, $k \geq 1$, which hopefully satisfy $x^{(k)} \approx \varphi(x^{(0)}, t^{(k)})$. Thus, the *phase variables* $x^{(k)}$ can be plotted against discrete time values $t^{(k)}$, and an approximation of the graph of $\varphi(x, t)$ with respect to t is obtained.

Quite often, however, one is often *not* interested in following the phase variables with respect to the time variable t, but rather with respect to another phase variable x_i. For example, it is very common that stopping conditions are expressed in terms of phase variables, and not the time variable.

In many physical applications and particularly in the theory of dynamical systems, one is often interested in computing a *Poincaré map* P (also known as a *first return map*) of a system such as (1). This map is produced by considering successive intersections of a trajectory with a codimension-one surface Σ—a Poincaré section—of the phase space $\mathbb{R}^n$.

[*] Tel.: +46-18-471-32-00; fax: +46-18-471-32-01.
E-mail address: warwick@math.uu.se (W. Tucker).

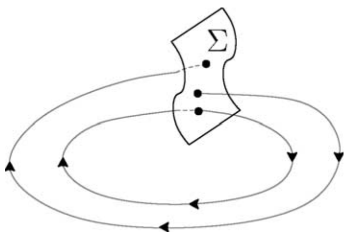

Fig. 1. The surface Σ and two trajectories.

Given a system (1), the existence of a Poincaré map is far from obvious, and in many cases it simply does not exist. One important occasion, however, where the Poincaré map *is* well defined is when the system admits periodic solutions. Indeed, let $x^{(0)}$ be a point on such a solution. Then there exists a positive number T, called the *period* of the orbit, such that $\varphi(x^{(0)}, T + t) = \varphi(x^{(0)}, t)$ for all $t \in \mathbb{R}$. In particular, $\varphi(x^{(0)}, T) = \varphi(x^{(0)}, 0) = x^{(0)}$, so the point $x^{(0)}$ returns to itself after having flowed for T time units. Now we consider a surface Σ that is transversal to the flow, i.e., the surface normal at $x^{(0)}$ satisfies $\langle n_\Sigma(x^{(0)}), f(x^{(0)})\rangle \neq 0$, where $\langle\cdot\rangle$ denotes the inner product. By the implicit function theorem we can find an open neighbourhood $U \subset \Sigma$ of $x^{(0)}$ such that for all $x \in U$, there exists a positive number $\tau(x)$ such that if $z = \varphi(x, \tau(x))$, then:

(a) $z \in \Sigma$ (x returns to the plane Σ after time $\tau(x)$);
(b) $\operatorname{sign}\langle n_\Sigma(x), f(x)\rangle = \operatorname{sign}\langle n_\Sigma(z), f(z)\rangle$ (Σ is passed from the same direction).

The function $\tau : \mathbb{R}^n \to \mathbb{R}_+$ is continuous, and represents the time it takes for the point x to return to Σ according to condition (b). The point $z = \varphi(x, \tau(x))$ is called the *first return* of x, and the Poincaré map $P : U \to \Sigma$ is defined by $P(x) = \varphi(x, \tau(x))$. Note that, by definition, we have $\tau(x^{(0)}) = T$ and $P(x^{(0)}) = x^{(0)}$ (Fig. 1).

The use of Poincaré maps reduces the study of flows to the study of maps—a topic that is more well understood, and therefore has a richer flora of theorems. It also reduces the dimension of the problem by 1: we may restrict our attention to points exclusively lying in the Poincaré section Σ. Unfortunately, except under the most trivial circumstances, the Poincaré map cannot be expressed by explicit equations. Instead, it is implicitly defined by the vector field f and the section Σ. Obtaining information about the Poincaré map thus requires both solving the system of differential equations, and detecting when a point has returned to the Poincaré section. In what follows, we will propose a numerical algorithm that addresses both requirements.

2. Classical numerical methods

As mentioned in Section 1, the approach usually taken consists of two parts: first one solves (1) using a suitable integration algorithm

$$t^{(k+1)} = t^{(k)} + \Delta t^{(k)}, \qquad x_i^{(k+1)} = x_i^{(k)} + \Delta t^{(k)} G_i(x^{(k)}, \Delta t^{(k)}) \quad (i = 1, \dots, n),$$

where, e.g. $G_i(x, h) = f_i(x + (h/2)f(x))$ in the case of the midpoint Euler method. This will produce a set of numerically computed grid points $(t^{(0)}, x^{(0)}), (t^{(1)}, x^{(1)}), \dots, (t^{(k)}, x^{(k)})$ along an approximate trajectory. There is a vast literature on this topic, so we shall not discuss the various choices of G.

Secondly, one tries to determine when the surface Σ is intersected. Since Σ is defined in terms of the phase variables $x_1, \dots, x_n$ and not in terms of the independent variable t, this is not an entirely trivial task. One can, however, easily detect when the surface has been crossed. Assuming[1] that Σ is defined by $\{x \in \mathbb{R}^n : x_{i_0} = C\}$, one simply has to check for a sign change in the quantity $C - x_{i_0}^{(k)}$. Also, as one is only interested in returns to the surface flowing in the same direction as the initial point, one wants the signs of $\langle n_\Sigma(x^{(0)}), f_{i_0}(x^{(0)})\rangle$ and $\langle n_\Sigma(x^{(k)}), f_{i_0}(x^{(k)})\rangle$ to coincide. Once such a crossing of Σ has been detected, the actual intersection $P(x^{(0)})$ can be approximated by an interpolation scheme, e.g. a bisection- or a Newton algorithm. Such an iterative ending may require many vector field evaluations, as well as a diminishing step-size Δt. This complicates any attempt to make a rigorous, global error estimate. Furthermore, as pointed out in [1], it is at this final stage the largest computational error usually occurs.

3. A new approach

Following the spirit of Hénon [1], we will transform one of the dependent variables $x_1, \dots, x_n$ into an independent one. As described in the above-mentioned paper, this provides a set-up allowing us to make the final approach to Σ in one single step, thereby avoiding the error accumulation associated with classical methods. The novelty of our approach is that, instead of applying this transformation only at the last stage, we will always ensure that one of the phase variables is independent. Not only does this take care of the problematic last step described above, it also facilitates a very powerful adaptive control which will be described later.

Given a point $x \in \mathbb{R}^n$, consider the dominating component of the vector field

$$|f_{\hat{\imath}}(x)| = \max_{i=1,\dots,n} \{|f_i(x)|\}.$$

Assuming that we are bounded away from fixed points of the system, the dominating component will always be bounded from below by a positive constant. Therefore, we can divide all components of the vector field by the number $f_{\hat{\imath}}(x)$. Using the notation $x_{n+1} = t$, we first embed the original system (1) into an $(n+1)$-dimensional system

$$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}t} = 1, \qquad \frac{\mathrm{d}x_i}{\mathrm{d}t} = f_i(x) \quad (i = 1, \dots, n). \tag{2}$$

By dividing each component of (2) by $f_{\hat{\imath}}(x)$, we get the rescaled system

$$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_{\hat{\imath}}} = \frac{1}{f_{\hat{\imath}}(x)}, \qquad \frac{\mathrm{d}x_i}{\mathrm{d}x_{\hat{\imath}}} = \frac{f_i(x)}{f_{\hat{\imath}}(x)} \quad (i = 1, \dots, n). \tag{3}$$

We shall from here on use the notation $\dot{x}$ and x' to denote differentiation with respect to t and $x_{\hat{\imath}}$, respectively.

Note that since $x_{\hat{\imath}}' = 1$, the variable $x_{\hat{\imath}}$ now is independent in (3). On the other hand, the formerly independent variable x_{n+1} has now become dependent. In view of this, we should take integration steps in terms of $\Delta x_{\hat{\imath}}$

$$x_{\hat{\imath}}^{(k+1)} = x_{\hat{\imath}}^{(k)} + \Delta x_{\hat{\imath}}^{(k)}, \qquad x_i^{(k+1)} = x_i^{(k)} + \Delta x_{\hat{\imath}}^{(k)} G_i(x^{(k)}, \Delta x_{\hat{\imath}}^{(k)}) \quad (i \in \{1, \dots, n+1\} \setminus \{\hat{\imath}\}).$$

Here, we would take

$$G_i(x, h) = \frac{f_i(x + (h/2) f(x)/f_{\hat{\imath}}(x))}{f_{\hat{\imath}}(x + (h/2) f(x)/f_{\hat{\imath}}(x))}$$

in the case of the midpoint Euler method. Applying such an integration scheme will now produce a sequence of $(n+1)$-dimensional grid points $x^{(1)}, x^{(2)}, \dots, x^{(k)}$, where we can explicitly specify the phase space step-size $\Delta x_{\hat{\imath}}^{(k)} = x_{\hat{\imath}}^{(k+1)} - x_{\hat{\imath}}^{(k)}$.

[1] The more general situation where Σ is defined by $S(x) = 0$ can be treated with minor modifications.

This integration procedure can be continued as long as the $\hat{\imath}$th component of the original vector field does not vanish. For topological reasons, however, this degenerate situation *must* occur if we are to return to Σ. When approaching such an instance, we simply change the independent coordinate according to the rule

$$|f_{\hat{\imath}}(x)| = \max_{i=1,\dots,n}\{|f_i(x)|\}.$$

By our assumption that we stay away from fixed points, there is always at least one non-zero vector field component at any given point $x \in \mathbb{R}^n$. Therefore the rescaled vector field (3) is always well defined.

The algorithmic implementation is actually quite simple. Using a high-level approach, the main stages can be described as follows.

Algorithm 1.

- Input: the vector field $f(x)$ and initial value $x^{(0)}$.

```
0: Set $k = 0$.
1: At the point $x^{(k)}$, compute the dominating coordinate: $\hat{\imath} = \hat{\imath}(x^{(k)})$.
2: Set $H(x) = 1/f_{\hat{\imath}}(x)$, and consider the system
       $\frac{dx_{n+1}}{dx_{\hat{\imath}}} = H(x), \quad \frac{dx_i}{dx_{\hat{\imath}}} = H(x)f_i(x) \quad (i = 1,\dots,n).$
3: Select a step-size $\Delta x_{\hat{\imath}}^{(k)}$. If this step-size will cause the trajectory to cross $\Sigma$, shorten it so that $x^{(k+1)}$ will
   land exactly on $\Sigma$.
4: Taking the step-size derived in Step 3, compute the next integration point $x^{(k+1)}$ using the system derived
   in Step 2.
5: If we do not have a return, set $k = k + 1$, and return to Step 1.
```

- Output $x^{(k+1)}$.

Recall that we are assuming that Σ is defined by $\{x \in \mathbb{R}^n : x_{i_0} = C\}$. To simplify matters further, we are also assuming that all first returns to Σ have the same dominating coefficient $\hat{\imath} = i_0$. This second assumption covers the most common applications, and allows us to make the exposition clear. A more general situation can naturally be handled, but at the expense of a more complicated algorithm. For example, if the variable x_{i_0} defining Σ is not dominating at the final stage, we can always force the algorithm to take a Δx_{i_0} step, which will complete the return to Σ.

4. Adaptive control

A question that naturally arises is how to choose the step-size $\Delta x_{\hat{\imath}}^{(k)}$ appearing in Step 3 of Algorithm 1. Of course, we cannot give a completely satisfactory answer to this question without resorting to the theory of auto-validating algorithms. This would involve a move from computing with floating point numbers to computer representable intervals, where all arithmetic operations are carried out with directed rounding. A nice exposition of such methods is given in, e.g. [4]. Remaining in the realm of ordinary floating point computations, we will nevertheless attempt to give a partial answer to the posed question. Let us introduce the following notation:

$$|f_{\hat{\jmath}}(x)| = \max_{i\in\{1,\dots,n\}\setminus\{\hat{\imath}\}}\{|f_i(x)|\}.$$

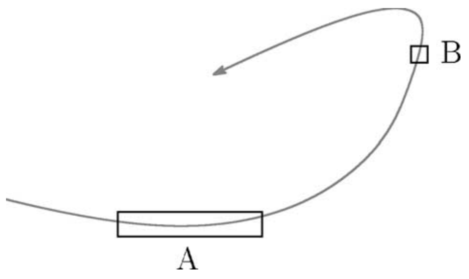

Fig. 2. Region A: $|f_{\hat{\imath}}(x)| \gg |f_{\hat{\jmath}}(x)|$; region B: $|f_{\hat{\imath}}(x)| \approx |f_{\hat{\jmath}}(x)|$.

The coordinate $\hat{\jmath}$ is thus the second largest component of f. We can now give a qualitative estimate of the step-size in terms of $f_{\hat{\imath}}$ and $f_{\hat{\jmath}}$: if $|f_{\hat{\imath}}(x^{(k)})| \gg |f_{\hat{\jmath}}(x^{(k)})|$, we should be able to take $\Delta x_{\hat{\imath}}^{(k)}$ relatively large. This is because at the point $x^{(k)}$, the flow is moving almost exclusively in the $x_{\hat{\imath}}$-direction. In other words, in a neighbourhood of $x^{(k)}$, the remaining components of the trajectory are almost constant. If, on the other hand, the two components $|f_{\hat{\imath}}(x^{(k)})|$ and $|f_{\hat{\jmath}}(x^{(k)})|$ are of the same order of magnitude, this is no longer the case. Instead, in a neighbourhood of $x^{(k)}$, the trajectory is arching at a considerable angle in the $(x_{\hat{\imath}}, x_{\hat{\jmath}})$-plane. In order to remain on the arch, we must now choose $\Delta x_{\hat{\imath}}^{(k)}$ relatively small, see Fig. 2.

But how large is "relatively large", and how small is "relatively small"? Again, we cannot expect to give a complete answer to this question. There are, however, some rules of thumb that are useful in most situations. First, we note that none of the n first components of the system (3) has an absolute value greater than 1. The last variable (the former time variable), however, may have a large modulus. Even if this happens to be the case, this particular variable is special in that it does not affect the other variables. Thus, if we are not too concerned about computing an accurate flow time, we can use the fact that the remaining components of the vector field are bounded by 1. Bearing this in mind, we may introduce the following quantity:

$$\Gamma(x) = \frac{|f_{\hat{\jmath}}(x)|}{|f_{\hat{\imath}}(x)|},$$

which satisfies $\Gamma(x) \in [0, 1]$ for all $x \in \mathbb{R}^n$. If we predefine Δ^- and Δ^+ to be the smallest resp. largest allowed step sizes, one natural choice would be

$$|\Delta x_{\hat{\imath}}^{(k)}| = \Gamma(x_{\hat{\imath}}^{(k)})\Delta^- + (1 - \Gamma(x_{\hat{\imath}}^{(k)}))\Delta^+.$$

Here Δ^- and Δ^+ would depend on the global error that is acceptable. In addition, Δ^- should also depend on the machine precision available. Both constants will also depend on p, the order of the integration method used in Step 4 of Algorithm 1. Indeed, assuming that the selected method is of order p, and that the step-size $|\Delta x_{\hat{\imath}}^{(k)}|$ is chosen to be Δ at every integration step, the global error $E_\Delta(x_{\hat{\imath}}^{(k)})$ can be bounded from above according to

$$E_\Delta(x_{\hat{\imath}}^{(k)}) \le \frac{\varepsilon(\Delta)}{K_{\hat{\imath}}}(\mathrm{e}^{K_{\hat{\imath}} \sum_k |x_{\hat{\imath}}^{(k)}|} - 1),$$

see, e.g. [2]. Here $K_{\hat{\imath}}$ is the Lipschitz constant of $f/f_{\hat{\imath}}$

$$K_{\hat{\imath}} = \min\left\{ \tilde{K} : \left\| \frac{f(x)}{f_{\hat{\imath}}(x)} - \frac{f(y)}{f_{\hat{\imath}}(y)} \right\| \le \tilde{K}\|x - y\| \right\},$$

and $\varepsilon(\Delta)$ denotes the slope error, which is known to satisfy the bound $\varepsilon(\Delta) \le C\Delta^p$. The constant C depends on the particular integration method we choose, and involves bounds of the vector field $f/f_{\hat{\imath}}$ and its derivatives.

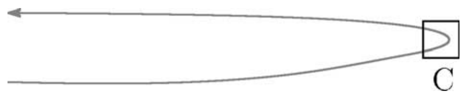

Fig. 3. Region C: a forbidden transition of type $+\hat{\imath} \to -\hat{\imath}$.

Following a trajectory from Σ until its full return, one will travel only a finite distance, so $\sum_k |x_{\hat{\imath}}^{(k)}|$ is bounded by some constant S. Therefore the global error at the return can be estimated by

$$E(\Delta) \leq \frac{C\Delta^p}{K_{\hat{\imath}}}(\mathrm{e}^{K_{\hat{\imath}}S} - 1) = \tilde{C}\Delta^p.$$

It follows that the step-size Δ should be proportional to the pth root of the maximally acceptable global error. Of course, we know little about the constant $\tilde{C}$ without more information regarding the explicit nature of the vector field f.

An easy way to detect a too large step-size is the following: whenever the dominating coordinate $\hat{\imath}$ changes, we make sure that the transition

$$(\hat{\imath}, \hat{\jmath}) \to (\hat{\jmath}, \hat{\imath})$$

takes place. If this is not the case, we may suspect that we are progressing too far in the dominating direction, i.e., our step-size is too large. The motivation for this requirement is simply one of continuity: all trajectories must be continuous, and so the first and second largest components of the vector field along a trajectory must also change in a continuous fashion.

In order to detect a more subtle situation, we may equip the dominating coordinate $\hat{\imath}$ with a $\pm$ sign, reflecting the sign of the associated vector field component. With this extended notation, we should decrease the step-size whenever a transition $\pm\hat{\imath} \to \mp\hat{\imath}$ is detected. This situation would indicate that the trajectory just made a very sharp 180° turn within one step. Clearly, this is not an acceptable situation, see Fig. 3.

5. Computing partial derivatives

In many situations we are interested in not only the propagation of a single initial point x, but rather a whole neighbourhood of x in Σ. Such information allows us to make predictions about the future of x, incorporating a limited amount of uncertainty in its measurement. In almost any physical application, this is very desirable, if not an absolute necessity. This prompts us to compute the matrix of partial derivatives $\mathrm{D}P$ of the Poincaré map P. Once we know $\mathrm{D}P$, we can compute its eigenvalues and eigenvectors λ_i and v_i, $i \in \{1, \ldots, n\} \setminus \{\hat{\imath}\}$, respectively. When classifying periodic orbits, we distinguish between three cases: (1) if an eigenvalue satisfies $|\lambda_i| < 1$, then we say that the direction v_i is *contracting* under the return map P; (2) if $|\lambda_i| > 1$, the direction v_i is *expanding*; (3) if $|\lambda_i| = 1$, the direction v_i is *neutral*. In terms of stability, an expanding direction v_i indicates that errors made in that same direction will be magnified by the factor $|\lambda_i|$ at each return. Unfortunately, we seldom can predict in what directions the initial errors are distributed. Thus the existence of *at least* one expanding direction suffices to render the orbit unstable. If, on the other hand, *all* directions are contracting, the orbit is said to be stable. Indeed, any reasonably small initial error will be contracted on its return.

From a local point of view, a Poincaré map P can be thought of as a composition of maps between very close codimension-one surfaces, $\Pi^{(k)} : \Sigma^{(k)} \to \Sigma^{(k+1)}$, where we define $\Sigma^{(0)} = \Sigma$. If the two surfaces $\Sigma^{(k)}$ and $\Sigma^{(k+1)}$ are at a distance d from each other, we can interpret the local Poincaré map $\Pi^{(k)}$ as a distance d map. Recall that we denote the flow of (1) by $\varphi(x,t)$. Thus, if we assume that $x^{(k)} \in \Sigma^{(k)}$, we may define a local Poincaré map $\Pi^{(k)} : \Sigma^{(k)} \to \Sigma^{(k+1)}$ by

$$\Pi^{(k)}(x^{(k)}) = \varphi(x^{(k)}, \tau(x^{(k)}, \Sigma^{(k+1)})),$$

where $\tau(x^{(k)}, \Sigma^{(k+1)})$ is the time it takes for $x^{(k)}$ to flow between $\Sigma^{(k)}$ and $\Sigma^{(k+1)}$. Here we are keeping with our strategy to always flow in the dominating direction of the flow. We will now provide an algorithm for computing the partial derivatives of the local Poincaré map $\Pi^{(k)}$. Once we have the ability to compute these local maps, we can form the partial Poincaré maps $P^{(k)} : \Sigma^{(0)} \to \Sigma^{(k)}$ by a simple series of compositions: $P^{(k)}(x) = \Pi^{(k-1)} \circ \cdots \circ \Pi^{(0)}(x)$. In what follows, we will suppress the superscript (k) for clarity.

Consider the partial derivatives of Π

$$\begin{aligned}\frac{\partial \Pi_i}{\partial x_j}(x) &= \frac{\partial}{\partial x_j}[\varphi_i(x,\tau(x))] = \frac{\partial \varphi_i}{\partial x_j}(x,\tau(x)) + \frac{\partial \tau}{\partial x_j}(x)\frac{\mathrm{d}\varphi_i}{\mathrm{d}t}(x,\tau(x)) = \frac{\partial \varphi_i}{\partial x_j}(x,\tau(x)) + \frac{\partial \tau}{\partial x_j}(x) f_i(\varphi(x,\tau(x))) \\ &= \frac{\partial \varphi_i}{\partial x_j}(x,\tau(x)) + \frac{\partial \tau}{\partial x_j}(x) f_i(\Pi(x)) \quad (i,j = 1,\dots,n). \end{aligned} \tag{4}$$

The partial derivatives of $\tau(x)$ are obtained by noting that $\Pi_{\hat{\imath}}(x)$ is constant. Thus

$$0 = \frac{\partial \Pi_{\hat{\imath}}}{\partial x_j}(x) = \frac{\partial \varphi_{\hat{\imath}}}{\partial x_j}(x,\tau(x)) + \frac{\partial \tau}{\partial x_j}(x) f_{\hat{\imath}}(\Pi(x)) \quad (j = 1,\dots,n),$$

and solving for $\partial\tau/\partial x_j$ yields

$$\frac{\partial \tau}{\partial x_j}(x) = -[f_{\hat{\imath}}(\Pi(x))]^{-1} \frac{\partial \varphi_{\hat{\imath}}}{\partial x_j}(x,\tau(x)) \quad (j = 1,\dots,n).$$

Inserting this expression into (4) gives

$$\frac{\partial \Pi_i}{\partial x_j}(x) = \frac{\partial \varphi_i}{\partial x_j}(x,\tau(x)) - \frac{\partial \varphi_{\hat{\imath}}}{\partial x_j}(x,\tau(x)) \frac{f_i(\Pi(x))}{f_{\hat{\imath}}(\Pi(x))} \quad (i,j = 1,\dots,n). \tag{5}$$

Note that the components with $i = \hat{\imath}$ vanish, just as desired.

Since we have already computed $\Pi(x)$, we can easily evaluate the right-most factor in (5). We have also computed the flow time $\tau(x)$ as our extended variable x_{n+1}. The partial derivatives of the flow still require some work, though. First, we need the differential equations for the partial derivatives. These are attained simply by differentiating the equations for the flow, $(\mathrm{d}/\mathrm{d}t)\varphi_i(x,t) = f_i(\varphi(x,t))$ $(i = 1,\dots,n)$, and changing the order of differentiation. On component level, this gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \varphi_i}{\partial x_j}(x,t) = \sum_{k=1}^{n} \frac{\partial f_i}{\partial x_k}(\varphi(x,t)) \frac{\partial \varphi_k}{\partial x_j}(x,t) \quad (i,j = 1,\dots,n), \tag{6}$$

or, in matrix form, $(\mathrm{d}/\mathrm{d}t)\,\mathrm{D}\varphi(x,t) = \mathrm{D}f(\varphi(x,t))\,\mathrm{D}\varphi(x,t)$, with the initial condition $\mathrm{D}\varphi(x,0) = I$, where I is the identity matrix. This is a linear differential equation, and is thus readily solved by standard numerical methods. Having done this, we simply insert the obtained values into the expression (5). This gives us the approximation of the partial derivatives of the local Poincaré map.

Returning to the task of computing the derivatives of the partial Poincaré maps $P^{(k)}$, it is simply a matter of multiplying the matrices $\mathrm{D}\Pi^{(k)}$ in the correct order

$$\mathrm{D}P^{(k)}(x) = \mathrm{D}\Pi^{(k-1)}\,\mathrm{D}\Pi^{(k-2)} \cdots \mathrm{D}\Pi^{(0)}(x).$$

This information can be updated at each integration step by the rules $\mathrm{D}P^{(0)}(x) = I$ and $\mathrm{D}P^{(k+1)}(x) = \mathrm{D}\Pi^{(k)}\,\mathrm{D}P^{(k)}(x)$, $k \geq 0$.

We can summarize the entire process by the following algorithm.

Algorithm 2.

- Input: the vector field $f(x)$ and initial value $x^{(0)}$.

```
0: Set $k = 0$, $\mathrm{D}X^{(0)} = I$, and compute $\hat{\imath} = \hat{\imath}(x^{(0)})$.
1: Set $H(x) = 1/f_{\hat{\imath}}(x)$, and consider the system
       $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_{\hat{\imath}}} = H(x), \qquad \frac{\mathrm{d}x_i}{\mathrm{d}x_{\hat{\imath}}} = H(x) f_i(x) \quad (i = 1, \dots, n).$
2: Select a step-size $\Delta x_{\hat{\imath}}^{(k)}$. If this step-size will cause the trajectory to cross $\Sigma$, shorten it so that $x^{(k+1)}$ will
   land exactly on $\Sigma$.
3: Taking the step-size derived in Step 2, compute the next integration point $x^{(k+1)}$ using the system derived
   in Step 1.
4: Using the local flow time $\Delta t^{(k)} = x_{n+1}^{(k+1)} - x_{n+1}^{(k)}$, compute the matrix $\mathrm{D}\varphi(x^{(k)}, \Delta t^{(k)})$ with $\mathrm{D}\varphi(x^{(k)}, 0) = I$
   from the system (6).
5: Recompute the dominating coordinate at the point $x^{(k+1)}$: $\hat{\imath} = \hat{\imath}(x^{(k+1)})$.
6: Compute $\mathrm{D}\Pi^{(k)}(x^{(k)})$ according to (5).
7: Update the matrix of accumulated partial derivatives:
       $\mathrm{D}X^{(k+1)} = \mathrm{D}\Pi^{(k)}(x^{(k)})\,\mathrm{D}X^{(k)}.$
8: If we do not have a return, set $k = k + 1$, and return to Step 1.
```

- Output $x^{(k+1)}$, $\mathrm{D}X^{(k+1)}$.

Let us briefly comment on a few differences between Algorithms 1 and 2. We have now added the computation of the matrices of partial derivatives of the partial Poincaré maps $\mathrm{D}P^{(k)}(x)$. The approximations are denoted $\mathrm{D}X^{(k)}$. Note that we also update the dominating coordinate $\hat{\imath}$ just before computing $\mathrm{D}X^{(k+1)}$. This is because the two consecutive local planes $\Sigma^{(k)}$ and $\Sigma^{(k+1)}$ need not be parallel. Indeed, this will be the case whenever $\hat{\imath}(x^{(k)})$ and $\hat{\imath}(x^{(k+1)})$ differ. By using $\hat{\imath}(x^{(k+1)})$ in Step 6, we ensure that the correct partial derivatives are computed (recall that the elements on the $\hat{\imath}$th row of $\mathrm{D}\Pi^{(k)}$ vanish).

At this point, one may argue that the outlined algorithm is not faithful to our general approach, where we work with $x_{\hat{\imath}}$-derivatives rather than time-derivatives. It is, however, very easy to make the necessary arrangements required to meet our needs. Set $H(x) = 1/f_{\hat{\imath}}(x)$, and consider the system

$$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_{\hat{\imath}}} = H(x), \qquad \frac{\mathrm{d}x_i}{\mathrm{d}x_{\hat{\imath}}} = H(x) f_i(x) \quad (i = 1, \dots, n). \tag{7}$$

Let $\psi(x, s)$ denote the solution to the n first variables of (7), i.e.

$$\psi_i'(x, s) = H(\psi(x, s)) f_i(\psi(x, s)) \quad (i = 1, \dots, n).$$

Thus $\psi(x, s)$ is the image of the point $\psi(x, 0) = x$ after having flowed the distance s in the $\hat{\imath}$-coordinate. If we consider the matrix of all partial derivatives of ψ, we have

$$\mathrm{D}\psi(x, \Delta x_{\hat{\imath}}) = I + \int_0^{\Delta x_{\hat{\imath}}} D[H(\psi(x, s)) f(\psi(x, s))]\,\mathrm{D}\psi(x, s)\,\mathrm{d}s. \tag{8}$$

The partial derivatives of the local Poincaré map are now readily available

$$\frac{\partial \Pi_i}{\partial x_j}(x) = \frac{\partial \psi_i}{\partial x_j}(x, \Delta x_{\hat{\imath}}) - \frac{\partial \psi_{\hat{\imath}}}{\partial x_j}(x, \Delta x_{\hat{\imath}}) H(\Pi(x)) f_i(\Pi(x)) \quad (i, j = 1, \dots, n), \tag{9}$$

where $\Delta x_{\hat{\imath}} = \Pi_{\hat{\imath}}(x) - x_{\hat{\imath}}$.

Algorithm 3.

- Input: the vector field $f(x)$ and initial value $x^{(0)}$.

0: Set $k = 0$, $DX^{(0)} = I$, and compute $\hat{\imath} = \hat{\imath}(x^{(0)})$.
1: Set $H(x) = 1/f_{\hat{\imath}}(x)$, and consider the system

$$\frac{dx_{n+1}}{dx_{\hat{\imath}}} = H(x), \qquad \frac{dx_i}{dx_{\hat{\imath}}} = H(x) f_i(x) \quad (i = 1, \dots, n).$$

2: Select a step-size $\Delta x_{\hat{\imath}}^{(k)}$. If this step-size will cause the trajectory to cross Σ, shorten it so that $x^{(k+1)}$ will land exactly on Σ.
3: Taking the step-size derived in Step 2, compute the next integration point $x^{(k+1)}$ using the system derived in Step 1.
4: Using the local step-size $\Delta x_{\hat{\imath}}^{(k)} = x_{\hat{\imath}}^{(k+1)} - x_{\hat{\imath}}^{(k)}$, use (8) to compute the matrix $D\psi(x^{(k)}, \Delta x_{\hat{\imath}}^{(k)})$ with $D\psi(x^{(k)}, 0) = I$.
5: Recompute the dominating coordinate at the point $x^{(k+1)}$: $\hat{\imath} = \hat{\imath}(x^{(k+1)})$.
6: Compute $D\Pi^{(k)}(x^{(k)})$ according to (9).
7: Update the matrix of accumulated partial derivatives:

$$DX^{(k+1)} = D\Pi^{(k)}(x^{(k)})\, DX^{(k)}.$$

8: If we do not have a return, set $k = k + 1$, and return to Step 1.

- Output $x^{(k+1)}$, $DX^{(k+1)}$.

6. Examples

The ideas discussed in this paper have been implemented in a C++ program. Below, we present the outcome of a few examples.

Example 1 (The Volterra system). Consider the system of differential equations

$$\dot{x}_1 = +\alpha x_1 - \beta x_1 x_2, \qquad \dot{x}_2 = -\gamma x_2 + \delta x_1 x_2,$$

which models a simple predator–prey system. Here the parameters α, β, γ and δ are all positive numbers, and we are only concerned with non-negative solutions. The exact solutions are implicitly given by level curves of the function

$$F(x_1, x_2) = |x_1|^{\gamma} |x_2|^{\alpha} e^{-(\delta x_1 + \beta x_2)}. \tag{10}$$

Since all level curves of (10) in the first quadrant are closed, we should have $P(x) = x$ for all initial points $x \in \Sigma$, where we may choose $\Sigma = \{x \in \mathbb{R}^2 : x_{\hat{\imath}} = C\}$. Furthermore, since the solutions form a continuous family of closed curves, the only relevant partial derivative of the Poincaré map must satisfy

$$\frac{\partial P_i}{\partial x_i}(x) = 1 \quad (i \in \{1, 2\} \setminus \{\hat{\imath}\})$$

Table 1
The Volterra system with $x = (1.5, 14)$ and step-size $\Delta x = 2^{-8} \approx 3.9 \times 10^{-3}$

Method	$\|P(x^{(0)}) - x^{(0)}\|$	$\|(P_1)'_{x_1} - 1\|$	Flow time	Number of steps
Euler	3.21411E−2	6.25192E−2	2.44847E+1	8579
Midpoint Euler	4.11446E−5	2.92632E−4	2.44472E+1	8569
Runge–Kutta 4	2.05751E−10	9.20541E−8	2.44471E+1	8569

for all x. In Table 1, we present the results of our method with initial condition $x^{(0)} = (1.5, 14)$ and $\Sigma = \{x \in \mathbb{R}^2 : x_2 = 14\}$. This set-up corresponds to the outmost orbit in Fig. 4.

Example 2 (The Lorenz equations). The following system of differential equations:

$$\dot{x}_1 = -\sigma x_1 + \sigma x_2, \qquad \dot{x}_2 = \varrho x_1 - x_2 - x_1 x_3, \qquad \dot{x}_3 = -\beta x_3 + x_1 x_2, \tag{11}$$

was introduced in 1963 by Lorenz, see [3]. As a crude model of atmospheric dynamics, these equations led Lorenz to the discovery of sensitive dependence of initial conditions—an essential factor of unpredictability in many systems. Numerical simulations for an open neighbourhood of the classical parameter values $\sigma = 10$, $\beta = 8/3$ and $\varrho = 28$ suggest that almost all points in phase space tend to a strange attractor—*the Lorenz attractor*, see Fig. 5(a). Recently, the author rigorously proved that this was not only a numerical mirage, but that the system (11) indeed supports a strange attractor, see [5,6].

For non-standard parameters, however, the system no longer displays chaotic dynamics. Instead almost all trajectories tend to a single, attracting periodic orbit, see Fig. 5(b). In Table 2, we present the results of our method with initial condition

$$x^{(0)} = (16.21325444114593, -55.78140243373939, 249)$$

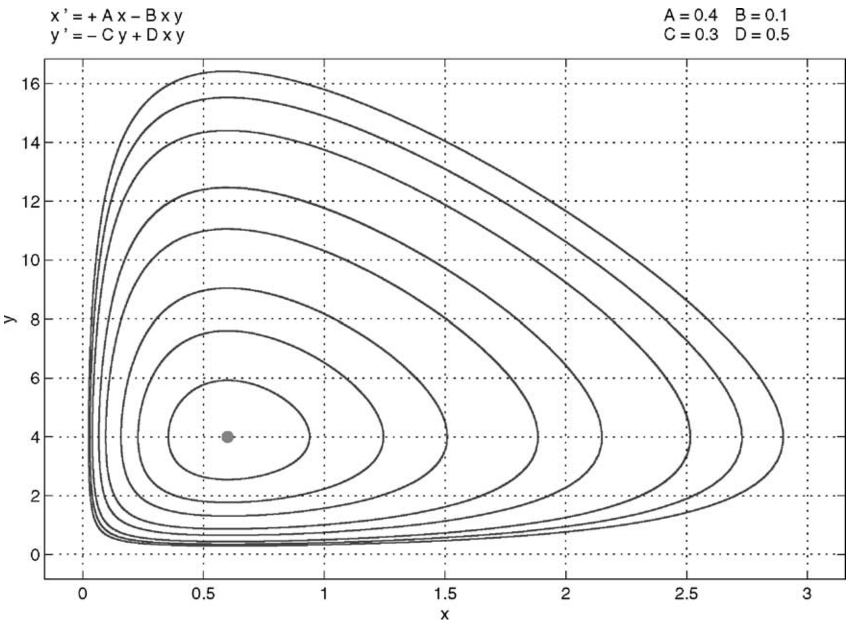

Fig. 4. A few closed orbits of the Volterra system.

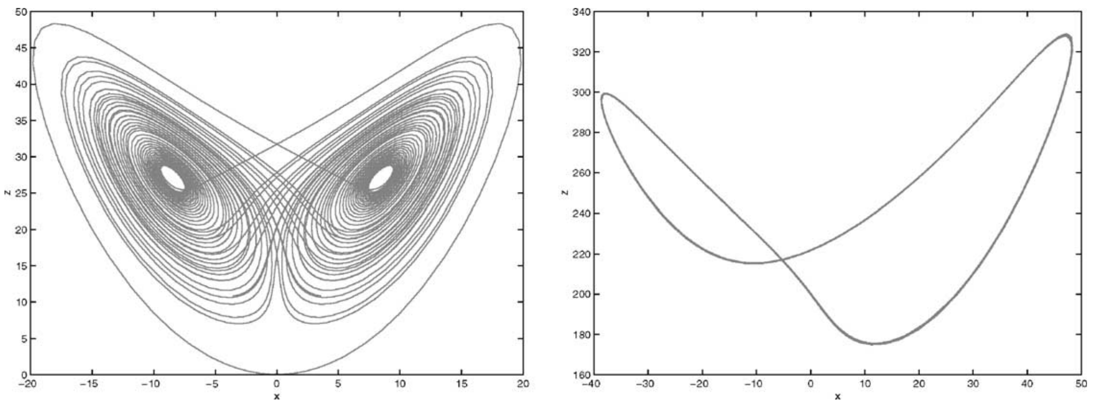

Fig. 5. The Lorenz system for (a) $r = 28$ and (b) $r = 250$.

Table 2
The Lorenz system with $(\sigma, \varrho, \beta) = (10, 250, 8/3)$ and step-size $\Delta x = 2^{-4}$

Method	$\|P(x^{(0)}) - x^{(0)}\|$	$\|\lambda_1 - \lambda_1^*\|$	$\|\lambda_2 - \lambda_2^*\|$	Flow time
Euler	4.71619E−1	2.08078E−4	1.01347E−2	4.587117410E−1
Midpoint Euler	6.22800E−5	5.12994E−7	1.71648E−5	4.600939402E−1
Runge–Kutta 4	2.15409E−10	2.43777E−12	9.68147E−11	4.600941506E−1

and $\Sigma = \{x \in \mathbb{R}^3 : x_3 = 249\}$. The surface Σ is pierced four times before the orbit re-connects with itself, so we had to modify the stopping condition accordingly. Simply looking at the partial derivatives of the Poincaré map is not very informative: instead we use $\mathrm{D}P(x^{(0)})$ to compute the corresponding eigenvalues λ_1 and λ_2. To get an idea of the convergence rate, we first computed the eigenvalues $(\lambda_1^*, \lambda_2^*) = (-6.217323621724878\text{E}{-3}, -2.989324529670951\text{E}{-1})$ using the Runge–Kutta method with step-size 2^{-10}. All subsequent computations were carried out with step-size $2^{-4} = 0.0625$.

The number of required integration steps were 9818 for the Euler method, and 9861 for both the midpoint Euler and the Runge–Kutta methods. Note that we indeed have contraction in both directions.

References

[1] M. Hénon, On the numerical computation of Poincaré maps, Physica D 5 (1982) 412–414.
[2] J.H. Hubbard, B.H. West, Differential Equations: A Dynamical Systems Approach, TAM 5, Springer, New York, 1991.
[3] E.N. Lorenz, Deterministic non-periodic flow, J. Atmos. Sci. 20 (1963) 130–141.
[4] R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1966.
[5] W. Tucker, The Lorenz attractor exists, C.R. Acad. Sci. Paris, Part 328, Sér. I (1999) 1197–1202.
[6] W. Tucker, A rigorous ODE solver and Smale's 14th problem, Found. Comput. Math. 2 (1) (2002) 53–117.