Computer assisted proof of chaos in the Lorenz equations[1]

Z. Galias, University of Mining and Metallurgy,
Department of Electrical Engineering,
al. Mickiewicza 30, 30-059 Kraków, Poland,
E-mail: galias@zet.agh.edu.pl

P. Zgliczyński,[2] Jagiellonian University,
Institute of Mathematics,
al. Reymonta 4, 30-059 Kraków, Poland,
E-mail: zgliczyn@im.uj.edu.pl

Abstract

In this paper we prove with computer assistance the existence of chaos in a suitable Poincaré map generated by the Lorenz system of equations. By chaos we mean the existence of symbolic dynamics with infinite number of periodic trajectories. The proof combines abstract results based on the fixed point index and finite rigorous computer calculations. Discussion concerning numerical algorithms is also included.

PACS codes: 0270, 0547, 0545

Keywords: chaos, computer assisted proof

[1]Research supported by Polish Scientific Grant no. 0449/P3/94/06 and by the University of Mining and Metallurgy, grant no. 10.120.132.

[2]corresponding author

0 Introduction

The Lorenz system of equations (1) introduced by Lorenz [L] is one of standard examples of deterministic chaos.

$$\begin{aligned} \dot{x} &= s(y-x), \\ \dot{y} &= rx - y - xz, \\ \dot{z} &= xy - qz, \end{aligned} \tag{1}$$

For various parameter values one can observe different behavior of solutions, bifurcations of periodic trajectories, horseshoes, strange attractors, etc. (see the book of Sparrow [Sp]). For parameter values $(s, r, q) = (10, 28, 8/3)$ one observe the famous *Lorenz attractor*. Picture of this attractor can be found in almost every modern book concerning dynamical systems or chaos, see for example book of Guckenheimer and Holmes [GH]. On the conceptual level the Lorenz attractor is well understood in terms of *geometrical models* [W], [ABS1], [ABS2], [Sh]. On the other hand a number of rigorous results concerning chaotic dynamics in the Lorenz system is rather small. The main difficulty is the inability to obtain estimates which show rigorously that assumptions of these models are satisfied. This difficulty is, of course, not unique to the Lorenz system. In fact, obtaining the necessary estimates is the central obstacle for most of non-linear analysis. In the result we report here the computer was used to overcome these difficulties.

The first rigorous results concerning chaos in Lorenz equations are *Hassard et all* [HHTZ] result for parameter values $(s, r, q) = (10, 76, 9)$ and *Mischaikow and Mrozek* [MM1] result for $(s, r, q) = (45, 54, 10)$. In these papers authors are able to show with computer assistance that some kind of symbolic dynamics is present, but the existence of periodic points is not claimed. There exists also an analytical result of Chen [Ch], who was able to establish that the Lorenz equations support a horseshoe, for large r and s close to $\frac{2q-1}{3}$. This is highly advantageous result. But as the other results cited before this result cannot be extended to the most popular values $(s, r, q) = (10, 28, 8/3)$.

In this paper we present the first proof that the Lorenz system with "classical" (most popular) parameter values $(s, r, q) = (10, 28, 8/3)$ has infinitely many qualitatively distinct periodic trajectories (see sec. 3 for a precise statement). This is done by showing rigorously with computer assistance that the second iterate of a suitably chosen Poincaré return map has a "topological horseshoe". In our approach, which is purely topological, we do not use any estimates concerning derivatives of the Poincaré map, which are necessary to establish a sort of hyperbolicity present in horseshoes [Mo]. This is obviously a weakness of the method, as we are unable to prove a sensitive dependence on initial conditions present in the horseshoe. Paradoxically, the topological character of this method decides about its strength, as assumptions of our topological theorem are relatively easy to check with computer assistance, which seems to be impossible for smooth methods with nowaday computers [SA].

Our method is a variation of the method for proving chaos in dynamical systems developed by the second author in [Z97b] and [Z96b]. This method was applied previously to the Rössler equations, the Hénon map and to the Chua's circuit [G]. In order to use this technique to prove the existence of chaos in Lorenz equations some modifications of the method were necessary. The method is based on the fixed point index and is similar to the Mischaikow and Mrozek method based on the discrete Conley index [MM1], [MM2]. On the topological level the differences between these methods are following:

- We replace the more general, but rather difficult concept of the discrete Conley index, by the fixed point index, which seems to be much easier concept. It is also particularly well suited to study fixed and periodic points.
- We define a class of TS-maps (see sec. 1) for which one can easily compute the fixed point indices of interest and we formulate theorem 1 about chaotic behavior for TS-map without any reference to the notion of the fixed point index.
- If the TS-map is a homeomorphism (for example it is a Poincaré map for ODE's) then assumptions of our theorem can be checked only on the boundaries, which has enormous impact on computation time (see sec. 5).

Both the methods (the one used in this paper and the method of Mischaikow and Mrozek) involve some algebraic topology and at first sight seem to be more complicated than the method of *Hassard et al.* [HHTZ] which is based on connectedness. Also the amount of computation for this last method is smaller. But we would like to stress that this method is less general, as it depends considerably on differential equations under investigations and cannot be used to prove the existence of periodic points.

The second aim of our paper is to show limitations of existing 'interval arithmetic' in the task of calculating in a reasonable time the image of 'big sets' with relatively poor accuracy. In the proof of chaos the evaluation of the Poincaré map is necessary. This involves integration of the system equations. The main problem encountered during integration of the dynamical system in 'interval arithmetic' is the 'wrapping effect' which causes very quick growth of the set of initial conditions for the next integration step. We overcome this problem by estimating the growth of error along the trajectory in the Euclidean norm while the set of initial conditions for integration method is kept very small (in fact we use a point interval). All the operations are performed in 'interval arithmetic' to obtain rigorous errors for elementary operations.

1 TS-maps

The aim of this section is to recall the notion of TS-maps introduced in [Z97a], which is special case of window chains introduced by R. Easton in [E75], [E89].

By $\mathbb{R}$, $\mathbb{R}_+$ $\mathbb{Z}$, $\mathbb{N}$ we will denote the sets of real numbers, nonnegative real numbers, integers and natural numbers (including zero) respectively. Let (X, ρ) be a metric space. Let $Z \subset X$. By $\mathrm{int}(Z)$, $\mathrm{cl}(Z)$, $\mathrm{bd}(Z)$ we denote respectively the interior, the closure and the boundary of the set Z.

Let $f : X \mapsto X$ be a continuous map and N be a subset of X. By $f_{|N}$ we will denote the map obtained by restriction of the domain of f to the set N. The maximum invariant part of N (with respect to f) is defined by

$$\mathrm{Inv}(N, f) = \bigcap_{i \in \mathbb{Z}} f_{|N}^{-i}(N).$$

For the union of disjoint rectangles $P = \bigcup P_k \subset \mathbb{R}^2$, where $P_k = [a_k, b_k] \times [c_k, d_k]$ we set

$$\mathrm{L}(P) := \bigcup \{a_k\} \times [c_k, d_k], \tag{2}$$

$$\mathrm{R}(P) := \bigcup \{b_k\} \times [c_k, d_k], \tag{3}$$

$$\mathrm{V}(P) := \mathrm{L}(P) \cup \mathrm{R}(P), \tag{4}$$

$$\mathrm{H}(P) := \bigcup ([a_k, b_k] \times \{c_k\} \ \cup \ [a_k, b_k] \times \{d_k\}). \tag{5}$$

So $\mathrm{L}(P)$, $\mathrm{R}(P)$, $\mathrm{V}(P)$, $\mathrm{H}(P)$ are unions of left vertical, right vertical, vertical and horizontal edges in P respectively.

In the remaining part of this section we consider maps on the plane $\mathbb{R}^2$.

Let us fix $u, d \in \mathbb{R}$, $u > d$ and a sequence $a_{-1} = -\infty < a_0 < a_1 < \ldots a_{2K-2} < a_{2K-1} < a_{2K} = \infty$, where $a_i \in \mathbb{R}$ for $i = 0, 1, \ldots, 2K-1$. Let

$$N_i := [a_{2i}, a_{2i+1}] \times [d, u], \quad \text{for} \quad i = 0, \ldots, K-1, \tag{6}$$

$$E_i := (a_{2i-1}, a_{2i}) \times [d, u], \quad \text{for} \quad i = 0, \ldots, K, \tag{7}$$

$$N := N_0 \cup N_1 \cup \ldots \cup N_{K-1}, \tag{8}$$

$$E := E_0 \cup E_1 \cup \ldots \cup E_{K-1} \cup E_K. \tag{9}$$

The sets E_i, N_i are contained in the horizontal strip $(-\infty, \infty) \times [d, u]$ in the following order (we compare x-coordinates)

$$E_0 < N_0 < E_1 < N_1 < \ldots < E_{K-1} < N_{K-1} < E_K. \tag{10}$$

Suppose further that for $i = 0, 1, \ldots, K$ we have sets E_i' such that $E_i \subset E_i'$, $\mathrm{cl}(E_i') \cap (\mathrm{H}(N) \setminus \mathrm{V}(N)) = \emptyset$, $\mathrm{cl}E_i' \cap \mathrm{cl}E_j' = \emptyset$ for $i \neq j$ and that there exist continuous homotopies $h_i : [0,1] \times E_i' \mapsto E_i'$ such that $h_i(0, p) = p$, $h_i(1, p) \in E_i$ and $h_i(t, p) = p$ for $t \in [0, 1]$, $p \in E_i$. This means that the set E_i' can be continuously deformed to the set E_i without any intersection with the set N. E_i is a *deformation retract* of E_i'. We set

$$E' := E_0' \cup E_1' \cup \ldots E_K'.$$

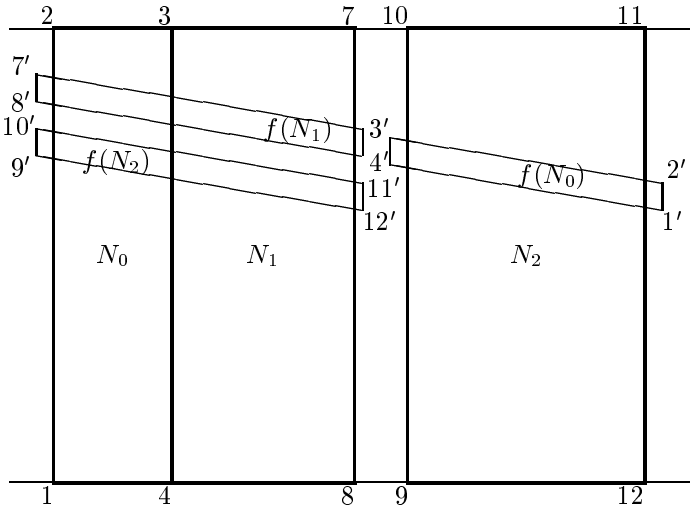

Figure 1: An example of TS-map. Sets N_0, N_1, N_2 and their images under f

Definition 1. *Let the sets E_i, E_i', N_i be as above. Let D be an open set such that $N \subset D$ and a map $f : D \mapsto \mathbb{R}^2$ be continuous. We say that f is* TS-*map (topological shift) (relatively to the sets N, E, E') if there exist functions $l, r : \{0, 1, \ldots, K-1\} \mapsto \{0, 1, \ldots, K\}$ such that the following conditions hold*

$$f(\mathrm{L}(N_i)) \subset E'_{l(i)}, \qquad f(\mathrm{R}(N_i)) \subset E'_{r(i)}, \tag{11}$$

$$f(N) \subset \mathrm{int}(E' \cup N). \tag{12}$$

Geometrically, the above conditions mean that the image of vertical edges does not intersect the set N and the image of N is contained in the set which can be continuously deformed to the horizontal strip without any intersection with horizontal edges of N.

If for some j we have $r(i) \le j < l(i)$ or $l(i) \le j < r(i)$ then we will say that the image of N_i *covers* N_j *horizontally*.

Figure 1 shows a schematic example of TS-map on three sets N_0, N_1, N_2. The sets E_i' coincide with E_i. The corners of N_i are denoted by numbers, their images by numbers with primes. Corners $5, 6$ are omitted because they almost coincide with corners $4, 3$ respectively. It is easy to see that the image of N_0 covers horizontally N_2 and both sets N_0, N_1 are covered horizontally by the images of N_1 and N_2.

We are looking for periodic points of the TS-map f. We will characterize them by periodic infinite sequences $c = (c_i)_{i \in \mathbb{N}}$ of symbols $0, 1, \ldots, K-1$ with the property $f^i(x) \in N_{c_i}$ for $i \in \mathbb{N}$.

Let $\Sigma_K := \{0,1,\ldots,K-1\}^{\mathbb{Z}}$, $\Sigma_K^+ := \{0,1,\ldots,K-1\}^{\mathbb{N}}$. Σ_K, Σ_K^+ are topological spaces with the Tichonov topology. On Σ_K, Σ_K^+ we have the shift map σ given by

$$(\sigma(c))_i = c_{i+1}.$$

Let $A = [\alpha_{ij}]$ be a $K \times K$-matrix, with nonnegative elements ($\alpha_{ij} \in \mathbb{R}_+ \cup \{0\}$ for $i,j = 0,1,\ldots,K-1$). We define $\Sigma_A \subset \Sigma_K$ and $\Sigma_A^+ \subset \Sigma_K^+$ by

$$\Sigma_A := \{c = (c_i)_{i\in\mathbb{Z}} \mid \alpha_{c_i c_{i+1}} > 0\}, \tag{13}$$

$$\Sigma_A^+ := \{c = (c_i)_{i\in\mathbb{N}} \mid \alpha_{c_i c_{i+1}} > 0\}. \tag{14}$$

Obviously Σ_A^+, Σ_A are invariant under σ.

Let f be a TS-map. To relate the dynamics of f on $\mathrm{Inv}(N,f)$ with shift dynamics on Σ_K^+ we introduce the *transition matrix of* f denoted by $A(f)$.

We define $A(f)_{ij}$, where $i,j = 0,1,\ldots,K-1$ by

$$A(f)_{ij} := \begin{cases} 1, & \text{if } E_{l(i)} < N_j < E_{r(i)} \text{ or } E_{l(i)} > N_j > E_{r(i)}\ , \\ 0, & \text{otherwise.} \end{cases}$$

It easy easy to see that $A(f)_{i,j} \neq 0$, if N_j lays between the images of vertical edges of N_i or in other words if $f(N_i)$ covers N_j horizontally (we deform the image by the homotopies h_i if necessary).

For $i \in \mathbb{N}$ we define the map $\pi_i : \mathrm{Inv}(N,f) \mapsto \{0,1,\ldots,K\}$ given by $\pi_i(x) = j$ iff $f^i(x) \in N_j$. Now we define the map $\pi : \mathrm{Inv}(N,f) \mapsto \Sigma_K^+$ by $\pi(x) := (\pi_i(x))_{i\in\mathbb{N}}$. The map π assigns to the point x the indices of sets N_i which its trajectory goes through. It is easy to see that we have

$$\pi \circ f = \sigma \circ \pi. \tag{15}$$

If f is a homeomorphism then the definition of π_i can be extended to all integers and in this case the domain of π is Σ_K.

Obviously the semi-conjugacy (15) alone is not a sign of complicated dynamics. It may happen that the set $\pi(\mathrm{Inv}(N,f))$ is finite or even empty. The dynamics is complicated if $\pi(\mathrm{Inv}(N,f))$ is infinite.

Definition 2. *Let $c \in \Sigma_K^+$ ($c \in \Sigma_K$). We will say that c is admissible for f in N if there exists $x_c \in N$ such that $f^i(x_c) \in N_{c_i}$ for $i \in \mathbb{N}$ ($\mathbb{Z}$). If c is periodic we additionally require that x_c is a periodic point with the same principal period as c.*

If $Z \subset \Sigma_K^+$ ($Z \subset \Sigma_K$) we will say that Z is admissible for f in N if every sequence in Z is admissible for f in N.

The following theorem, proved in [Z97a], gives the characterization of the set of admissible sequences for TS-maps.

Theorem 1. *Let f be a TS-map. Then $\Sigma^+_{A(f)} \subset \pi(Inv(N,f))$. The preimage of any periodic sequence from $\Sigma^+_{A(f)}$ contains periodic points of f. If we additionally suppose that f is a homeomorphism then $\Sigma_{A(f)} \subset \pi(Inv(N,f))$.*

Results similar to the theorem above, but without the existence of periodic points, can be found also in [MM2] and [E89].

2 TS-map structure for the Lorenz system

In this section we describe an application of theorem 1 to the case when the sets N_i are non-disjoint. This is the situation encountered by us in the Lorenz equations (see next section).

Let us fix $u, d \in \mathbb{R}$, $u > d$ and two sequences (a_{0i}), (a_{1i}) with elements $a_{ki} \in \mathbb{R} \cup \{-\infty, \infty\}$ for $i = -1, 0, 1, 2, 3, 4$ and $k = 0, 1$, such that for fixed k the following conditions hold $a_{k,-1} = -\infty < a_{k,0} < a_{k,1} < a_{k,2} < a_{k,3} < a_{k,4} = \infty$. Let us define

$$N_{ki} := [a_{k,2i}, a_{k,2i+1}] \times [d, u], \quad \text{for} \quad i = 0, 1, \tag{16}$$

$$E_{ki} := (a_{k,2i-1}, a_{k,2i}) \times [d, u], \quad \text{for} \quad i = 0, 1, 2, \tag{17}$$

$$N_k := N_{k0} \cup N_{k1}, \quad \text{for} \quad k = 0, 1, \tag{18}$$

$$N := N_0 \cup N_1, \tag{19}$$

$$E_k := E_{k0} \cup E_{k1} \cup E_{k2}, \tag{20}$$

$$E := E_0 \cup E_1. \tag{21}$$

Obviously for fixed k the sets N_{ki} are disjoint, but it may happen that for example $N_{0i} \cap N_{1i} \neq \emptyset$.

Let D be an open set, $N \subset D$ and $f : D \mapsto \mathbb{R}^2$ be a continuous map. Suppose that the following conditions hold (compare Fig. 3 in the next section)

$$f(N) \subset \mathrm{int}(E \cup N), \tag{22}$$

$$f(\mathrm{L}(N_{00})) \subset E_{11}, \quad f(\mathrm{R}(N_{00})) \subset E_{12}, \tag{23}$$

$$f(\mathrm{L}(N_{01})) \subset E_{10}, \quad f(\mathrm{R}(N_{01})) \subset E_{11}, \tag{24}$$

$$f(\mathrm{L}(N_{10})) \subset E_{00}, \quad f(\mathrm{R}(N_{10})) \subset E_{02}, \tag{25}$$

$$f(\mathrm{L}(N_{11})) \subset E_{00}, \quad f(\mathrm{R}(N_{11})) \subset E_{02}. \tag{26}$$

It should be noted that in the above conditions assumptions concerning the images of N_{ki} are expressed using the sets $E_{1-k,j}$.

Let $M \geq 0$ be a real number. We define

$$\tilde{N}_{0i} := N_{0i}, \tag{27}$$

$$\tilde{E}_{0i} := E_{0i}, \tag{28}$$
$$\tilde{N}_{1i} := N_{1i} + (M, 0), \tag{29}$$
$$\tilde{E}_{1i} := E_{1i} + (M, 0), \tag{30}$$
$$\tilde{N}_k := \tilde{N}_{k0} \cup \tilde{N}_{k1}, \tag{31}$$
$$\tilde{N} := \tilde{N}_0 \cup \tilde{N}_1. \tag{32}$$

We define $\tilde{f} : \tilde{N} \mapsto \mathbb{R}^2$ by

$$\tilde{f}(x) := \begin{cases} f(x) + (M, 0) & \text{for } x \in \tilde{N}_0, \\ f(x) - (M, 0) & \text{for } x \in \tilde{N}_1. \end{cases} \tag{33}$$

From compactness of N and continuity of f it follows that there exists M such that

$$\tilde{N}_0 \cap \tilde{N}_1 = \emptyset, \tag{34}$$
$$\tilde{f}(\tilde{N}_k) \cap \tilde{N}_k = \emptyset, \text{ for } k = 0, 1. \tag{35}$$

We fix such M. Hence from conditions (22–26) we get

$$\tilde{f}(\tilde{N}) \subset \text{int}(\tilde{E} \cup \tilde{N}), \tag{36}$$
$$\tilde{f}(\text{L}(\tilde{N}_{0,0})) \subset \tilde{E}_{1,1}, \qquad \tilde{f}(\text{R}(\tilde{N}_{0,0})) \subset \tilde{E}_{1,2}, \tag{37}$$
$$\tilde{f}(\text{L}(\tilde{N}_{0,1})) \subset \tilde{E}_{1,0} \cap \tilde{E}_{0,2}, \quad \tilde{f}(\text{R}(\tilde{N}_{0,1})) \subset \tilde{E}_{1,1}, \tag{38}$$
$$\tilde{f}(\text{L}(\tilde{N}_{1,0})) \subset \tilde{E}_{0,0}, \qquad \tilde{f}(\text{R}(\tilde{N}_{1,0})) \subset \tilde{E}_{0,2} \cap \tilde{E}_{1,0}, \tag{39}$$
$$\tilde{f}(\text{L}(\tilde{N}_{1,1})) \subset \tilde{E}_{0,0}, \qquad \tilde{f}(\text{R}(\tilde{N}_{1,1})) \subset \tilde{E}_{0,2} \cap \tilde{E}_{1,0}. \tag{40}$$

We will treat the indices of the sets $\tilde{N}_{ki}$ as binary expansions so 00 corresponds to 0, 01 to 1 , 10 to 2 and 11 to 3. With this convention we see that $\tilde{f}$ is a TS-map with a transition matrix $A(\tilde{f})$ given by

$$A(\tilde{f}) := \begin{bmatrix} 0, & 0, & 1, & 1 \\ 0, & 0, & 1, & 1 \\ 0, & 1, & 0, & 0 \\ 1, & 0, & 0, & 0 \end{bmatrix}.$$

From theorem 1 applied to the map $\tilde{f}$ it follows that the set $\Sigma^+_{A(\tilde{f})}$ is admissible for the map $\tilde{f}$. Let us observe that

$$A^2(\tilde{f}) := \begin{bmatrix} 1, & 1, & 0, & 0 \\ 1, & 1, & 0, & 0 \\ 0, & 0, & 1, & 1 \\ 0, & 0, & 1, & 1 \end{bmatrix}.$$

The following lemma follows from the form of the square of transition matrix of $\tilde{f}$ and the condition (35).

Lemma 2. *The set* $\{00,01\}^N \cup \{10,11\}^N$ *is admissible for* $\tilde{f}^2$.

There is an obvious connection between f^2 and $\tilde{f}^2$. From the construction of $\tilde{f}$ it follows immediately that

$$x \in N_0, f(x) \in N_1 \text{ iff } x \in \tilde{N}_0, \tilde{f}(x) \in \tilde{N}_1, \tag{41}$$

$$f^2(x) = \tilde{f}^2(x), \text{ if } x \in N_0, \quad f(x) \in N_1. \tag{42}$$

Similar statements with obvious modifications hold when we exchange indices 0 and 1 in the above conditions.

From lemma 2 and conditions (41), (42) we obtain

Theorem 3. *Let* f *and sets* N_{ki}, E_{ki} *be as above and conditions (36–40) are satisfied. Then the set* $\{00,01\}^{\mathbb{N}} = \Sigma_2^+$ *is admissible for* f^2 *in* $N_0 = (N_{00} \cup N_{01})$.

3 Chaos in the Lorenz equations

The Lorenz equations are given by [L]

$$\begin{aligned} \dot{x} &= s(y-x), \\ \dot{y} &= rx - y - xz, \\ \dot{z} &= xy - qz, \end{aligned} \tag{43}$$

where $s = 10$, $r = 28$, $q = 8/3$.

We consider a transversal plane $\Sigma = \{(x,y,z) \in \mathbb{R}^3 : z = r-1\}$. This a standard choice for the Poincaré section. Let $\mathbf{P}$ be a Poincaré map generated on the plane Σ, i.e., for $\mathbf{x} \in \Sigma$ by $\mathbf{P}(\mathbf{x})$ we denote the point at which the trajectory based at $\mathbf{x}$ intersects Σ for the first time in the specified direction.

We have found for $\mathbf{P}$ the structure described in the previous section. To express this structure we introduce the new rotated coordinates on the plane $z = r-1$

$$\bar{x} := x\cos\theta - y\sin\theta, \tag{44}$$

$$\bar{y} := x\sin\theta + y\cos\theta, \tag{45}$$

where the angle $\theta = 70^\circ$ ($\theta = 2\pi(70/360)$ in radians). The line $\bar{x} = 0$ is very close to the intersection of the stable manifold of the origin $(0,0,0)$ with the plane $z = r-1$ in the region of interest.

Let us set $a_{00} = -1.6$, $a_{01} = -0.4$, $a_{02} = 0.4$, $a_{03} = 1.6$, $a_{10} = -3.3$, $a_{11} = -0.9$, $a_{12} = 0.9$, $a_{13} = 3.3$, $d = -6$, $u = 6$. We define sets N_{ki}, E_{ki} as in the previous section. Sets N_{ki} are shown in the Fig. 2. Despite of the change of coordinates (44,45) we will use the notion of the left, right, vertical and horizontal edges with respect to the old coordinates in order to be consistent with the formulation of theorems in sections 1 and 2.

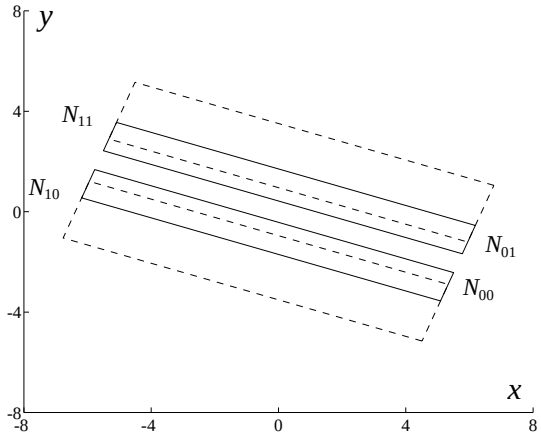

Figure 2: Rectangles N_{00}, N_{01}, N_{10} and N_{11} on the transversal plane. N_{00} and N_{01} are printed with solid lines, while N_{10} and N_{11} with dashed ones

In computer simulations we have integrated equations (43) using the fourth-order Runge-Kutta algorithm with the time step $h = 0.005$. In Fig. 3 we show the images of borders of N_{ki} under Poincaré map obtained by numerical integration. The image of N_{00} covers N_{11} horizontally and similarly the image of N_{01} covers N_{10} (compare Fig. 3a). Both images of N_{10} and N_{11} covers $N_{00} \cup N_{01}$ horizontally (see Fig. 3b). These results indicate that complex behavior of the system and existence of infinitely many periodic orbits is possible. The next step is to prove strictly this observation. In order to perform this task we have developed a computer program using procedures for interval computations from the BIAS and PROFIL packages [K].

With computer assistance we have proved the following lemma.

Lemma 4. *The Poincare map* $\mathbf{P}$ *is well defined and continuous on* N. *The conditions* (22–26) *hold for* $\mathbf{P}$.

Proof of this lemma will be given in the next section. Combining the above lemma and theorem 3 we obtain main theorem of this paper.

Theorem 5. *For all parameter values in a sufficiently small neighborhood of* $(s, r, q) = (10, 28, 8/3)$ *there exists a transversal section* $I \subset \{z = 27\}$ *such that the Poincaré map* $\mathbf{P}$ *induced by* (43) *is well defined and continuous on* I. *There exists continuous surjective map* $\pi : Inv(I, \mathbf{P}^2) \to \Sigma_2$, *such that*

$$\pi \circ \mathbf{P}^2 = \sigma \circ \pi.$$

The preimage of any periodic sequence from Σ_2 *contains periodic points of* $\mathbf{P}^2$.

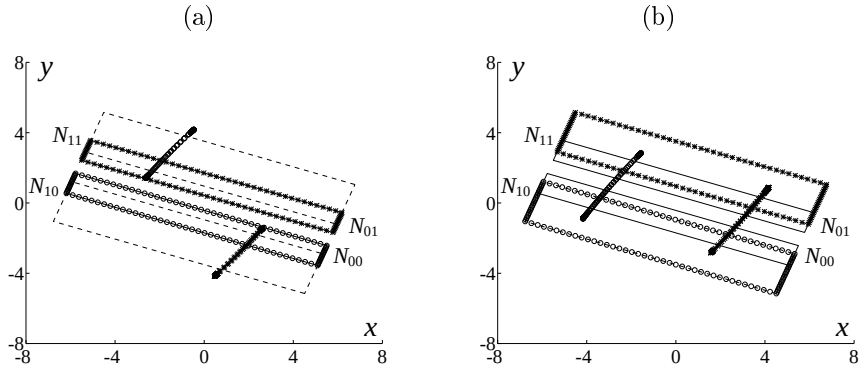

Figure 3: Images of borders of N_{00}, N_{01}, N_{10} and N_{11} on the transversal plane — computer simulations, (a) images of edges of N_{00} and N_{01}, one can clearly see that image of N_{00} covers N_{11} horizontally and symmetrically the image of N_{01} covers N_{10}, (b) images of edges of N_{10} and N_{11}, both images covers N_{00} and N_{01} horizontally

Proof. From continuous dependence of the solutions of ODE's on parameters it follows easily that lemma 4 holds in some neighborhood U of $(s, r, q) = (10, 28, 8/3)$ in the parameter space. We fix U and we consider the Poincaré map $\mathbf{P}$ generated by (43) for parameters values from U.

We set $I := \mathrm{Inv}(N_{00} \cup N_{01}, \mathbf{P}^2)$. We define the map $\pi : I \to \Sigma_2$ by

$$\pi_i(\mathbf{x}) = 0, \text{ if } \mathbf{P}^{2i}(\mathbf{x}) \in N_{00}, \tag{46}$$

$$\pi_i(\mathbf{x}) = 1, \text{ if } \mathbf{P}^{2i}(\mathbf{x}) \in N_{01}. \tag{47}$$

From theorem 3 we obtain that every periodic sequence from Σ_2 is admissible for $\mathbf{P}$ in $N_{00} \cup N_{01}$. Now from density of the periodic sequences in Σ_2 we get $\pi(I) = \Sigma_2$. □

4 Details of computer calculations

In our computer program we have used the procedures for interval computations from BIAS and PROFIL packages [K] prepared by Olaf Knüppel from Technical University Hamburg-Harburg.

In this section we present the detailed description of the procedure for computation of the image of a rectangle under Poincaré map and then we will describe the proof of lemma 4.

The whole computer program used during the computer-assisted proof can be found at:

<http://galaxy.uci.agh.edu.pl/~galias/int.html>.

4.1 One integration step

First we describe the procedure `NextPointExactTaylor4` for computation of the image of an Euclidean ball $\mathrm{B}(P_0, \varepsilon_0)$ under dynamical system after time h, where P_0 is a three-dimensional interval vector and ε_0 is a positive real value. The procedure finds a three-dimensional interval vector P_2 and a real value ε_2 such that the image of the ball $\mathrm{B}(P_0, \varepsilon_0)$ after time h is enclosed within the ball $\mathrm{B}(P_2, \varepsilon_2)$

$$\varphi(\mathrm{B}(P_0, \varepsilon_0), h) \subset \mathrm{B}(P_2, \varepsilon_2). \tag{48}$$

For that task the fourth-order Taylor integration formula and the logarithmic norm are used. They are described in the following subsections.

4.1.1 Procedure for computation of $\varphi(P, [0, h])$

Before we present the implementation of the Taylor integration formula and the computation of logarithmic norm let us describe the procedure `GetTraj`(P, P_T, h), which is used several times in the program. Its code is given in the Appendix. This procedure computes the three-dimensional interval P_T containing all the trajectories starting from the three-dimensional interval vector P after time $t \in [0, h]$. The procedure is based on the following lemma:

Lemma 6. *Let $\mathbf{y}(0)$ be a set of initial conditions. Let Y be a convex subset of $\mathbb{R}^3$, ε be a positive real number and let us define $Y_\varepsilon := \mathrm{B}(Y, \varepsilon)$. By $\mathrm{Hull}(A)$ we will denote the smallest closed ball in the max-norm containing the set A. Let*

$$X := \mathbf{y}(0) + [0, h] \cdot \mathbf{f}(\mathbf{y}(0)) + \frac{[0, h]^2}{2} \mathrm{Hull}(\mathbf{f}'(Y_\varepsilon)\mathbf{f}(Y_\varepsilon)),$$

where all operations on the right side are of set type, for example $\mathbf{f}(Y_\varepsilon) = \{\mathbf{x} : \mathbf{x} = \mathbf{f}(\mathbf{y})$, for some $\mathbf{y} \in Y_\varepsilon\}$ and $[0, h] \cdot \mathbf{f}(\mathbf{y}(0)) = \{t \cdot \mathbf{x} : t \in [0, h], \mathbf{x} \in \mathbf{f}(\mathbf{y}(0))\}$.

If $X \subset Y$ then $\mathbf{y}([0, h]) := \varphi(\mathbf{y}(0), [0, h]) \subset X$.

Proof. Let $t = \inf\{s : \mathbf{y}(s) \notin X\}$. We will show that $t \geq h$. If $t < h$ then there exists $\delta > 0$ such that $\mathbf{y}(s) \in Y_\varepsilon$ for $t \leq s \leq t + \delta < h$. From the first order Taylor formula for all $s \in [t, t + \delta]$ we have for $i = 1, 2, 3$ $\mathbf{y}_i(s) = \mathbf{y}_i(0) + s\mathbf{f}_i(\mathbf{y}(0)) + \frac{1}{2}s^2(\mathbf{f}'(\mathbf{y})\mathbf{f}(\mathbf{y}))_i$, where $\mathbf{y} \in Y_\varepsilon$ depends on s and i. It follows that $\mathbf{y}(s) \in X$ for all $s \in [t, t + \delta]$. Hence $\mathbf{y}(s) \in X$ for all $s \in [0, t + \delta]$ and $\inf\{s : \mathbf{y}(s) \notin X\} \geq t + \delta > t$ which is a contradiction. □

In the procedure `GetTraj`(P, P_T, h) we first choose Y. Then using the first order Taylor formula ($\mathbf{y}(t + h) = \mathbf{y} + h\mathbf{y}'(t) + \frac{1}{2}h^2\mathbf{y}''(t + \lambda h)$) we compute image P_T of P after time $[0, h]$, where y'' is evaluated over the set $Y_\varepsilon \supset Y$. If the image is enclosed in Y then P_T is returned. Otherwise we choose a bigger set Y and repeat the computations. From the previous lemma it follows that the following proposition is true.

Proposition 1. *If* `GetTraj`(P, P_T, h) *returns* `TRUE` *then* $\varphi(P, [0, h]) \subset P_T$.

4.1.2 Taylor integration method

For integration of the equation (43) the fourth-order Taylor formula has been used. We have also tested the fourth-order Runge-Kutta formula but the computation time was longer due to greater number of coefficients in the exact formula for the error term. Let us denote by $\mathbf{y}(t) = (y_1(t), y_2(t), y_3(t))^T$ the solution of the equation (43). Let us recall that the standard fourth-order Taylor integration method is based on the expansion

$$y_i(t+h) = y_i(t) + hy_i'(t) + \frac{1}{2}h^2 y_i''(t) + \frac{1}{6}h^3 y_i'''(t) + \frac{1}{24}h^4 y_i^{(4)}(t) + \frac{1}{120}h^5 y_i^{(5)}(t+\lambda_i h), \tag{49}$$

where $i = 1, 2, 3$ and $\lambda_i \in [0, 1]$ for $i = 1, 2, 3$. Using the equation $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ we can easily compute $\mathbf{y}^{(k)}$ in terms of $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$. For the Lorenz system

$$\mathbf{f}'(\mathbf{y}) \cdot \mathbf{h} = \begin{pmatrix} -S & S & 0 \\ R - y_3 & -1 & -y_1 \\ y_2 & y_1 & -Q \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}, \tag{50}$$

where $\mathbf{y} = (y_1, y_2, y_3)^T$ and $\mathbf{h} = (h_1, h_2, h_3)^T$. The second derivative $\mathbf{f}''(\mathbf{y})$ does not depend on $\mathbf{y}$ and can be computed as

$$\mathbf{f}''(\mathbf{y}) \cdot (\mathbf{h}_1, \mathbf{h}_2) = \begin{pmatrix} 0 \\ -h_{11}h_{32} - h_{31}h_{12} \\ h_{11}h_{22} + h_{21}h_{12} \end{pmatrix}, \tag{51}$$

where $\mathbf{h}_1 = (h_{11}, h_{21}, h_{31})$ and $\mathbf{h}_2 = (h_{12}, h_{22}, h_{32})$. As the left hand $\mathbf{f}(\mathbf{y})$ of the equation (43) does not contain terms of order greater than two it is clear that $\mathbf{f}^{(k)} \equiv 0$ for $k > 2$. $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ are computed within the program using procedures `LeftSide`, `FPrim` and `FBis` given in the Appendix. `FBis2` is the `FBis` procedure for the case of two equal arguments.

Using the chain rule of differentiation we can obtain formulas for $\mathbf{y}^{(k)}$. In the following we write $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ instead of $\mathbf{f}(\mathbf{y}(t))$, $\mathbf{f}'(\mathbf{y}(t))$ and $\mathbf{f}''(\mathbf{y}(t))$ respectively.

$$\begin{aligned}
\mathbf{y}'(t) &= \mathbf{f}(\mathbf{y}(t)) = \mathbf{f}, \\
\mathbf{y}''(t) &= \frac{d}{dt}(\mathbf{f}(\mathbf{y}(t))) = \frac{d\mathbf{f}}{dt}(\mathbf{y}(t))\frac{d\mathbf{y}}{dt}(t) = \mathbf{f}'(\mathbf{y}(t))\mathbf{f}(\mathbf{y}(t)) = \mathbf{f}'\mathbf{f}, \\
\mathbf{y}'''(t) &= \mathbf{f}''\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}, \\
\mathbf{y}^{(4)}(t) &= 3\mathbf{f}''\mathbf{f}'\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}, \\
\mathbf{y}^{(5)}(t) &= 4\mathbf{f}''\mathbf{f}\mathbf{f}''\mathbf{f}\mathbf{f} + 4\mathbf{f}''\mathbf{f}'\mathbf{f}'\mathbf{f}\mathbf{f} + 3\mathbf{f}''\mathbf{f}'\mathbf{f}\mathbf{f}'\mathbf{f} + 3\mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f}'\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}.
\end{aligned}$$

The formula we use for computation of $\mathbf{y}(t+h)$ reads

$$\mathbf{y}(t+h) = \mathbf{y}(t) + h\mathbf{f} + \frac{h^2}{2}\mathbf{f}'\mathbf{f} + \frac{h^3}{6}(\mathbf{f}''\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}) + \frac{h^4}{24}(3\mathbf{f}''\mathbf{f}'\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f} + \mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}) + e(\mathbf{y}, h), \tag{52}$$

where $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ stands for $\mathbf{f}(\mathbf{y}(t))$, $\mathbf{f}'(\mathbf{y}(t))$ and $\mathbf{f}''(\mathbf{y}(t))$. The error $e(\mathbf{y}, h)$ introduced by omitting the higher order terms is equal to

$$e(\mathbf{y}, h) = \frac{h^5}{120}(4\mathbf{f}''\mathbf{f}\mathbf{f}''\mathbf{f}\mathbf{f}+4\mathbf{f}''\mathbf{f}'\mathbf{f}'\mathbf{f}\mathbf{f}+3\mathbf{f}''\mathbf{f}'\mathbf{f}\mathbf{f}'\mathbf{f}+3\mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f}'\mathbf{f}+\mathbf{f}'\mathbf{f}'\mathbf{f}''\mathbf{f}\mathbf{f}+\mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}), \quad (53)$$

where this time $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ are computed at points $y_i(t + \lambda_i h)$ with $\lambda_i \in [0, 1]$.

4.1.3 Logarithmic norm

The procedure `LogNorm`(P, L) computes the upper bound L of logarithmic norm of the matrix $\mathbf{f}'(\mathbf{y})$ over the three-dimensional interval vector P. Let us recall that the Logarithmic norm [HNW] of matrix Q is defined by

$$m(Q) := \lim_{h \to 0, h > 0} \frac{||I + hQ|| - 1}{h}. \quad (54)$$

For the Euclidean norm on the right side of the above equation the logarithmic norm of Q can be obtained using the formula

$$m(Q) = \text{ largest eigenvalue of the matrix } \frac{1}{2}(Q^T + Q). \quad (55)$$

In the procedure `LogNorm(P,L)` we first compute the coefficients of the characteristic equation of the matrix $(\mathbf{f}'(P) + \mathbf{f}'(P)^T)/2$. Then using the Cardano formula we find roots of the characteristic equation and we choose the largest one (the roots are real as the matrix $(\mathbf{f}'(P) + \mathbf{f}'(P)^T)/2$ is symmetric). If for some reason this computation is not possible then `LogNorm` returns `FALSE`. If the procedure returns `TRUE` then L is the upper bound of the logarithmic norm over the set P.

4.1.4 Procedure `NextPointExactTaylor4`

The procedure `NextPointExactTaylor4`$(P_0, \varepsilon_0, P_2, \varepsilon_2)$ computes the image of a ball $\mathrm{B}(P_0, \varepsilon_0)$ under dynamical system after time h. For given 3D interval vector P_0 and real value ε_0 the procedure finds a 3D interval vector P_2 and a real value ε_2 such that the image of the ball $\mathrm{B}(P_0, \varepsilon_0)$ after time h is enclosed within the ball $\mathrm{B}(P_2, \varepsilon_2)$

$$\varphi(\mathrm{B}(P_0, \varepsilon_0), h) \subset \mathrm{B}(P_2, \varepsilon_2). \quad (56)$$

If the procedure is not capable to compute the image it returns `FALSE`. In the opposite case it computes P_2 and ε_2 and returns `TRUE`. The procedure consists of two parts.

In the first part the image of P_0 after time h is computed using the fourth-order Taylor integration formula with exact computation of the error term. We first calculate P_2 according to equation (52) without the last term $e(\mathbf{y}, h)$. During this computation $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ are computed over the set P_0. Then using the procedure

`GetTraj`(P_0, h) we find the set Y, containing trajectories starting from P_0 after time $t \in [0, h]$:

$$\varphi(P_0, [0, h]) \subset Y.$$

Then we compute $e(P_0, h)$ using equation (53), where $\mathbf{f}$, $\mathbf{f}'$ and $\mathbf{f}''$ are computed over Y. Finally we modify P_2 by adding the error term and we obtain a three-dimensional interval vector P_2 containing the image of P_0 after time h, i.e.,

$$\varphi(P_0, h) \subset P_2.$$

In the second part of the procedure we compute the change of the radius of the Euclidean ball during evolution after time h. This computation is based on the following theorem.

Theorem 7 ([HNW]). *Suppose that* $\mathbf{v}(t)$ *and* $\mathbf{w}(t)$ *are solutions of the system of differential equations* $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ *satisfying* $||\mathbf{v}(t_0) - \mathbf{w}(t_0)|| \leq \varepsilon$. *Let us also assume that the logarithmic norm* $m(\mathbf{f}'(\mathbf{y})) \leq L$ *on the convex set containing trajectories* $\{\mathbf{v}(t)\colon t \in [t_0, t_1]\}$ *and* $\{\mathbf{w}(t)\colon t \in [t_0, t_1]\}$. *Then for* $t \in [t_0, t_1]$ *we have the estimate*

$$||\mathbf{w}(t) - \mathbf{v}(t)|| \leq \varepsilon \mathrm{e}^{L(t-t_0)}. \tag{57}$$

In order to use the above theorem we have to compute the logarithmic norm over the convex set containing all the trajectories starting from $\mathrm{B}(P_0, \varepsilon_0)$ after time $[0, h]$. In order to find such a convex set we call the procedure `GetTraj` with the parameters $\mathrm{B}(P_0, \varepsilon_0)$ and h obtaining $Y \supset \varphi(\mathrm{B}(P_0, \varepsilon_0), h)$ (as Y is an interval vector it is obviously convex). Then we call the procedure `LogNorm`(Y, L) obtaining the upper bound L of the logarithmic norm of the matrix $\mathbf{f}'(\mathbf{y})$ over the set Y and finally we increase the size of the ball according to the following formula:

$$\varepsilon_2 = \varepsilon_0 e^{Lh}. \tag{58}$$

From the considerations presented above it follows that

Proposition 2. *If the procedure* `NextPointExactTaylor4`$(P_0, \varepsilon_0, P_2, \varepsilon_2)$ *returns* `TRUE` *then*

$$\varphi(\mathrm{B}(P_0, \varepsilon_0), h) \subset \mathrm{B}(P_2, \varepsilon_2).$$

In order to minimize wrapping effect the procedure `NextPointExactTaylor4` is called with parameter P_0 being a three-dimensional point interval.

Although P_0 is a point interval P_2 is an interval vector with nonzero diameter due to the existence of the error term in the integration formula and computation errors. Before calling the procedure `NextPointExactTaylor4` again the interval P_2 is shrinked to the point and ε_2 is increased appropriately. In this way we do not control the size of error by the interval arithmetic methods. Instead we use explicitly the Lipschitz constant obtained using the logarithmic Euclidean norm. Such action reduces the wrapping effect, which would cause very quick growth of computational errors in case of using interval arithmetic alone.

4.1.5 Reduction of wrapping effect with logarithmic norm

We have also tested other possibilities of computation of the image of an interval vector under dynamical system.

The first possibility tested was the one without logarithmic norm. This solution appeared to be much worse. Due to the wrapping effect the diameter of the interval grows much quicker than in the case when we use the logarithmic norm.

In order to show how big the difference is let us denote by d_1 the diameter of the rectangle which image under Poincaré map we compute and by d_2 the diameter of the rectangle returned by the procedure for evaluation of the Poincaré map. When we used logarithmic norm the quotient d_2/d_1 was between 70 and 490 for $d_1 = 0.005$, while in the second case it was greater then 9×10^9 for $d_1 = 5\times10^{-9}$ (for greater d_1 we were even not able to evaluate the Poincaré map). With logarithmic norm the computation time was approximately 7 hours. It was estimated to be more than 10^7 times longer without logarithmic norm.

We have also tried to use the logarithmic norm based on maximum norm instead of Euclidean norm. In this case the quotient d_2/d_1 was approximately 1000 times greater than in the case of logarithmic norm based on Euclidean norm.

Computation method	d_2/d_1	Computation time
logarithmic norm based on Euclidean norm	$70-490$	7h
logarithmic norm based on maximum norm	$>10^5$	$>10^4$h
without logarithmic norm	$>10^9$	$>10^8$h

Table 1: Comparison of computation time for different methods

These results show that without logarithmic norm based on Euclidean norm we would not be able to prove the assumptions about Poincaré map (compare also table 1).

4.2 Procedure for the Poincaré map

Once we have the procedure for computation of one integration step we can construct procedure for the whole Poincaré map $\mathbf{P}$. The procedure `PoincFun` computes image of the two-dimensional interval P_{start} contained in the transversal plane under Poincaré map. It returns a two-dimensional interval P_{end} such that

$$\mathbf{P}(P_{\text{start}}) \subset P_{\text{end}}.$$

During the procedure we perform subsequent integration steps calling procedure `NextPointExactTaylor4` obtaining balls $\mathrm{B}(P_n, \varepsilon_n)$ containing images $\varphi(P_{\text{start}}, nh)$ of the initial rectangle after h, $2h$,.... Initially we assign P_{start} to P_0 and set $\varepsilon_0 = 0$ (hence $\mathrm{B}(P_0, \varepsilon_0) = P_{\text{start}}$). Before every integration step we shrink the three-dimensional interval vector P_n to the point interval vector and increase ε_n

appropriately. This task is performed by the procedure `DecPIncEpsilon` (see Appendix), which finds a point interval vector P_n and a real ε_n such that

$$\mathrm{B}(P_{n,\mathrm{old}}, \varepsilon_{n,\mathrm{old}}) \subset \mathrm{B}(P_n, \varepsilon_n).$$

This action is necessary in order to avoid the wrapping effect.

The result of integration is a sequence of pairs P_n, ε_n fulfilling conditions

$$\varphi(\mathrm{B}(P_n, \varepsilon_n), h) \subset \mathrm{B}(P_{n+1}, \varepsilon_{n+1}), \quad P_{\mathrm{start}} = \mathrm{B}(P_0, \varepsilon_0).$$

A trajectory of a point in $N_{00} \cup N_{10}$ intersects the transversal plane twice before we can evaluate the Poincaré map. The first intersection is in a different direction. During the procedure the position of the trajectory is constantly monitored. As the image of the initial rectangle is within the ball $\mathrm{B}(P_n, \varepsilon_n)$ an intersection is not finished until the whole ball lies on the proper side of the transversal plane $\{z = r-1\}$. From the beginning of the procedure we check for the first intersection. If $\mathrm{B}(P_n, \varepsilon_n) \subset \{(x, y, z)\colon z > r - 1\}$ the boolean variable `FirstSection` is set to `TRUE`, which means that the first intersection has already been finished. Then we search for the beginning of the second section. We look for the smallest n such that $\mathrm{B}(P_n, \varepsilon) \cap \{(x, y, z)\colon z < r - 1\} \neq \emptyset$. At this moment we start to compute the interval vector `PPoincFull` containing the image of the initial rectangle under Poincaré map. We assign it to be `PPoincFull` $= P_{n-1}$. In every iteration the three-dimensional interval `PPoincFull` is increased, it becomes a convex hull of the previous value of `PPoincFull` and the set $\varphi(\mathrm{B}(P_{n-1}, \varepsilon_{n-1}), [0, h])$. Integration is continued until the second intersection with the transversal plane is finished. This corresponds to the first integration step for which trajectory lies completely after transversal plane ($\mathrm{B}(P_n, \varepsilon_n) \subset \{z < r - 1\}$).

The image of the initial rectangle under Poincaré map is contained in the projection of `PPoincFull` to the transversal plane.

In the course of the procedure we constantly check the transversality condition in order to ensure that the trajectory does not intersect the transversal plane more than twice before the image under Poincaré map is evaluated and that intersections with transversal plane are really transversal. In fact it should be checked only three times. First time at the beginning of the procedure to ensure that the trajectory enters the half-space $\{z < r - 1\}$ (the condition is $z'(P_{\mathrm{start}}) < 0$). The second time it should be checked during the first intersection: for each n such that $\varphi(\mathrm{B}(P_n, \varepsilon_n), [0, h]) \cap \{(x, y, z)\colon z = r - 1\} \neq \emptyset$ one should check if $z'(\varphi(\mathrm{B}(P_n, \varepsilon_n), [0, h])) > 0$. The third time it should be checked during the second intersection. This time we should check if $z'(\varphi(\mathrm{B}(P_n, \varepsilon_n), [0, h])) < 0$. In order to simplify the procedure and to shorten the computation time we perform the transversality check in the procedure `NextPointExactTaylor4`. For every n such that

$$\varphi(\mathrm{B}(P_n, \varepsilon_n), [0, h]) \cap \{(x, y, z)\colon z = r - 1\} \neq \emptyset$$

we check whether

$$z'(\varphi(\mathrm{B}(P_n, \varepsilon_n), [0, h])) \neq 0.$$

If this condition does not hold then `NextPointExactTaylor4` returns `FALSE`. From the discussion presented above it follows that

Proposition 3. *If the procedure* `PoincFun`$(P_{\rm start}, P_{\rm end})$ *returns* `TRUE` *then*

$$\mathbf{P}(P_{\rm start}) \subset P_{\rm end}.$$

4.3 Computer assisted proof

Using the procedure `PoincFun` we have performed a computer-assisted proof of the following theorem.

Theorem 8. *For all parameter values in a sufficiently small neighborhood of* $(S, R, Q) = (10, 28, 8/3)$

1. *there exists a continuous Poincaré map defined on* $N_{00} \cup N_{10}$,
2. *images of 'horizontal' edges* $\mathrm{H}(N_{00} \cup N_{10})$ *of* N_{00} *and* N_{10} *lie in the interior of* $N \cup E$ *i.e.,*
   - $\mathbf{P}(\mathrm{H}(N_{00} \cup N_{10})) \subset \mathrm{int}(N \cup E)$,
3. $\mathbf{P}(N_{00})$ *covers* N_{11} *horizontally and* $\mathbf{P}(N_{10})$ *covers* $N_{00} \cup N_{10}$ *horizontally, i.e., images of 'vertical' edges of* N_{00} *and* N_{10} *fulfill the following conditions*
   - $\mathbf{P}(\mathrm{L}(N_{00}))$ *lies on the left side of* N_{11} *i.e.,* $\mathbf{P}(\mathrm{L}(N_{00})) \subset E_{11}$,
   - $\mathbf{P}(\mathrm{R}(N_{00}))$ *lies on the right side of* N_{11} *i.e.,* $\mathbf{P}(\mathrm{R}(N_{00})) \subset E_{12}$,
   - $\mathbf{P}(\mathrm{L}(N_{10}))$ *lies on the left side of* N_{00} *i.e.,* $\mathbf{P}(\mathrm{L}(N_{10})) \subset E_{00}$,
   - $\mathbf{P}(\mathrm{R}(N_{10}))$ *lies on the right side of* N_{01} *i.e.,* $\mathbf{P}(\mathrm{R}(N_{10})) \subset E_{02}$.

Proof. During the proof we have used the procedure `PoincFun` for evaluation of the Poincaré map.

1. In the first step the set $N_{00} \cup N_{10}$ was covered by 56970 rectangles. We computed images of these rectangles under Poincaré map proving in this way the existence of continuous Poincaré map.
2. 'Horizontal' edges of $N_{00} \cup N_{10}$ were covered by 70 rectangles each. We proved that images of all of these rectangles lie within the strip $\mathrm{int}(N \cup E)$. In Fig. 4.a one can see the rectangles covering the bottom horizontal edge of $N_{00} \cup N_{10}$ and the rectangles containing their images under Poincaré map computed with the procedure `PoincFun`.

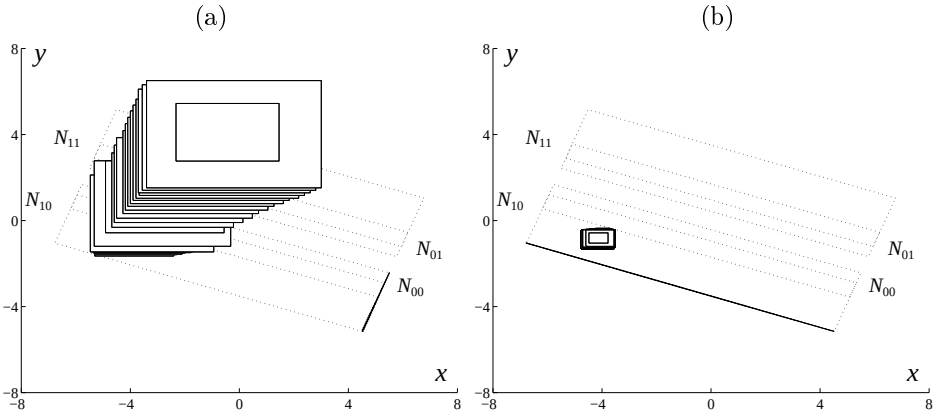

Figure 4: Images of edges of N_{ij} under Poincaré map — exact computations, (a) 70 rectangles covering the bottom vertical edge of $N_{00} \cup N_{10}$ and their images computed with the procedure `PoincFun` (it has been checked that the output rectangles lie in $\mathrm{int}(N \cup E)$, (b) 320 rectangles covering $\mathrm{L}(N_{10})$ and their images computed with the procedure `PoincFun` (it has been checked that the output rectangles lie in the set E_{02})

3. 'Vertical' edges $\mathrm{L}(N_{00})$, $\mathrm{R}(N_{00})$, $\mathrm{L}(N_{10})$ and $\mathrm{R}(N_{10})$ were covered by 351, 7744, 320 and 1177 rectangles respectively. For each of these rectangles the procedure `PoincFun` was called. We proved that images of these rectangles are included within appropriate subsets of $E \cup N$. An example of covering of an horizontal edge is shown in Fig. 4.b. In this figure one the covering of $\mathrm{L}(N_{10})$ with rectangles and images of these rectangles under Poincaré map computed with the procedure `PoincFun` are shown. One can see that output rectangles lie in E_{02}.

□

In the first part of the proof we have used the time step $h = 0.01$, while in the second and third parts we have used $h = 0.003$. In the first part we have used greater time step as we were not interested in the size of the rectangles returned by the procedure and hence we could accept greater errors. The time interval over which we had to integrate the equations to evaluate the Poincaré map varied from 0.67 to 0.94 for different points within $N_{00} \cup N_{10}$. This corresponds to less then 320 integration steps for the evaluation of the Poincaré map when using the time step equal to 0.003. The ratio of the size of the output rectangle to the size of the initial rectangle on which the Poincaré map is evaluated depends on the position of the initial rectangle and varies from 27 to 530.

All the computations were performed using the double precision — this is the precision implemented in the BIAS and PROFIL packages. We believe however

that the accuracy does not have much influence on the performance of the method. We think that for smaller accuracy (single precision or even less) it would also be possible to carry out the proof. In this case we would have to divide the sets under consideration into smaller parts. This would increase the computation time.

The above proof was performed on the Sun Ultra 1 computer, with 167 MHz clock. The program was compiled with gnu C++ compiler. It took approximately 7 hours to complete the proof.

The above theorem is not strictly equivalent to Lemma 4. Instead of proving the condition $\mathbf{P}(N_{00} \cup N_{10}) \subset \mathrm{int}(N \cup E)$ we only proved this condition for the border, namely $\mathbf{P}(\mathrm{bd}(N_{00} \cup N_{10})) \subset \mathrm{int}(N \cup E)$ with an additional condition that the Poincaré map $\mathbf{P}$ is defined on the whole $N_{00} \cup N_{10}$. From that one can prove that condition $\mathbf{P}(N_{00} \cup N_{10}) \subset \mathrm{int}(N \cup E)$ holds (compare proof of lemma 4 below). Such a proceeding reduces the computational time considerably. When we check only the existence of the Poincaré map we do not have to worry about the size of the images of rectangles under Poincaré map. Hence we can choose bigger rectangles covering $N_{00} \cup N_{10}$. During the proof of the first part of theorem 8 we needed 56970 rectangles. If we would check the stronger condition we would need approximately 10 times more rectangles, which would increase the computation time.

Proof of lemma 4. Due to symmetry of the problem from theorem 8 it follows that the Poincaré map $\mathbf{P}$ is well defined and continuous on N and conditions (23–26) are satisfied. It remains to show that from

$$\mathbf{P}(\mathrm{bd}(N_{ki})) \subset \mathrm{int}(N \cup E), \tag{59}$$

for $k, i = 0, 1$ the condition (22) follows.

Let us define $W := \mathrm{int}(N \cup E)$. The set W is the horizontal strip i.e. $W = (-\infty, \infty) \times (d, u)$. From the general theory of differential equations it follows that the Poincaré map $\mathbf{P}$ is a homeomorphism onto the image. From the Jordan-Brouwer Theorem [Gr, Th. 18.6, 18.7, 18.8] the set $\mathbf{P}(\mathrm{int}N_{ki})$ is open and

$$\mathrm{bd}(\mathbf{P}(\mathrm{int}N_{ki})) = \mathbf{P}(\mathrm{bd}(N_{ki})). \tag{60}$$

Now suppose that for some k, i the set $\mathbf{P}(N_{ki})$ is not contained in W. From (59) and definition of W it follows that there exists $\mathbf{x} \in \mathrm{int}(N_{ki})$ such that $\mathbf{P}(\mathbf{x})) \notin W$. Let $\mathbf{P}(\mathbf{x}) = (x, y)$. We may assume that $y \geq u$ (i.e., $\mathbf{P}(\mathbf{x})$ is above the strip W). Now consider the half-ray $H = \{\mathbf{P}(\mathbf{x}) + (0, t), \quad t \in \mathbb{R}_+\}$.

Obviously $H \cap \mathbf{P}(\mathrm{int}N_{ki}) \neq \emptyset$ and H is unbounded. So H is not contained in $\mathbf{P}(\mathrm{int}N_i)$ and from connectedness of H and (60) it follows that

$$H \cap \mathbf{P}(\mathrm{bd}(N_{ki})) \neq \emptyset. \tag{61}$$

From (59) and (61) it follows that $H \cap W \neq \emptyset$. But by construction $H \cap W = \emptyset$. Hence we get a contradiction. Thus (22) holds. □

5 Discussion of the relevance of the method in the computer assisted proofs

In the previous section we have proved the existence of symbolic dynamics for the second iterate of the Poincare map $\mathbf{P}$ by rigorous numerical calculation of $\mathbf{P}$, but not the second iterate. Another method to obtain this result would be to calculate $\mathbf{P}^2$ and then to apply the theorem 1 directly. Both approaches to this proof look almost equivalent, but the second one is much more time-consuming. We want now to explain this in detail.

Method 1. We have to calculate $\mathbf{P}$ on the sets N_{00}, N_{01}, N_{10}, N_{11} with error less then ϵ_1.

Method 2. We have to calculate $\mathbf{P}^2$ on the sets N_{00}, N_{01} only with error less then ϵ_2.

Below we define some quantities for both methods, we will index them by $m = 1, 2$ for the method 1 and 2 respectively.

To calculate the image of the edges of N_{ki} or the image of the entire set N_{ki} we cover it by the finite number of segments or rectangles with the size δ_m given by $\delta_m = \epsilon_m / L_m$, where L_m is the Lipschitz constant for the map $\mathbf{P}^m$. In the real algorithm we calculate the Lipschitz constants locally, but for our heuristic discussion we will assume that L_m is constant. The number of rectangles p_m covering edges of N_{ki} is given by $p_m = d_m/\delta_m = d_m L_m/\epsilon_m$, where d_m is the total length of edges of N_{ki} required in the method (for $m = 1$ the domain of calculations is bigger).

Now let t_m be a processor time required for the calculation of $\mathbf{P}^m(\mathbf{x})$, for $m = 1, 2$. This quantity depends on the point $\mathbf{x}$, but for convenience we will assume it is constant. The total time T_m required to calculate the image of the edges of N_{ki} is given by the formula

$$T_m = p_m t_m = d_m L_m t_m / \epsilon_m.$$

Let us assume that $L_2 = L_1^2$, $t_2 = 2t_1$, $\epsilon_1 = \epsilon_2$, $d_2 = 2d_1$. These assumptions are nearly fulfilled in our case. Using these assumptions we obtain

$$T_2/T_1 = L_1$$

for the calculation of image of edges.

In our calculations $L_1 \in (70, 500)$. This shows clearly the advantage of the first method.

Other important issues which differ our method from the method used by Mischaikow and Mrozek are following:

- we are able to reduce a significant part of calculations to the boundary of the sets under considerations (this reduce computation time almost 10 times), see sec. 4.3,
- we use Euclidean logarithmic norm, which is apparently harder to compute than the logarithmic norms based on max or sum (reduction factor of the order 10^5), see sec. 4.1.5, table 1,
- instead of Runge-Kutta method we use fourth-order Taylor method which produces twice smaller errors, see sec. 4.1.2,
- we use locally calculated Lipschitz constants and error bounds.

At this point one may wonder how can Mischaikow and Mrozek perform theirs calculations in a reasonable time, as they do not do any of the things listed above. The main point is that they invented method of intermediate sections, which is in spirit very close to the idea of the double cover described above and give similar reduction factor. They used 23 intermediate sections, for theirs choice of parameters to show that there is a topological horseshoe for the Poincare map. But for the classical parameter values they are unable to show that for the second iterate of Poincare map in a reasonable time.

We believe that further reduction of computation time is possible. It appears that substantial reduction of computation may be obtained by

- the use of Mischaikow and Mrozek intermediate section, but the evolution between those sections should be rather followed by our methods, which use the better norm for the problem,
- the enclosure of solution of ODE's can be probably better calculated using Lohner algorithm [Lo],
- the choice of the initial sets can be optimized and the domains for TS-maps can be generated by computer, as the work of Szymczak [Sz] shows.

The points listed above will obviously complicate considerably our simple numerical algorithms, which are relatively easy to implement. One should also have in mind, that an elaborate algorithm which theoretically gives better estimates can be so time consuming, that this can make it unusable. This is for example the case of higher order (> 4) Runge-Kutta or Taylor integration methods.

6 Appendix

6.1 General remarks

In the PROFIL package the following data types are defined: `REAL`, `INTERVAL`, `INTERVAL_VECTOR`, `INTERVAL_MATRIX`. The `REAL` data type is defined as the type

Operation	Return type	Description
`Inf( a )`	`REAL`	lower bound of `a`
`Sup( a )`	`REAL`	upper bound of `a`
`Hull( r )`	`INTERVAL`	point interval `[r,r]`
`Hull( x,y )`	`INTERVAL`	convex hull of `x` and `y`
`Succ( a )`	`INTERVAL`	smallest interval in which `a` is contained in the inner
`Mid( a )`	`REAL`	midpoint of `a`
`Diam( a )`	`REAL`	diameter of `a`
`Sqr( a )`	`INTERVAL`	square of `a`
`Intersection( a,b,c )`	`INT`	if `b` and `c` intersect 1 is returned and `a` contains the intersection, otherwise 0 is returned
`x <= a`		returns `TRUE` if `x` is contained in `a`
`x < a`		returns `TRUE` if `x` is contained in the interior of `a`
`Norm( iv )`	`INTERVAL`	computes the 2-norm of `iv`

Table 2: Operations and procedures from BIAS and PROFIL packages used in our program. `r` is of type `REAL`, `x`, `y` are of type `REAL` or `INTERVAL`, `a`, `b` and `c` are of type `INTERVAL`, `iv` is of type `INTERVAL_VECTOR`

`double`, the `INTERVAL` data type is a structure composed of two `REAL`s being the lower and upper bound of the interval. The `INTERVAL_VECTOR` is a vector of `INTERVAL`s (if `iv` is `INTERVAL_VECTOR` then `iv(i)` denoted the i^{th} element of `iv`. The `INTERVAL_MATIRX` is a matrix of `INTERVAL`s. In the BIAS package all the basic operations on the types specified above are implemented. This includes addition, subtraction, multiplication and division. For example expression like $C = A + B$, where A, B and C are intervals is implemented in such a way that

$$\{a + b : a \in A, b \in B\} \subseteq C.$$

In the table 2 we describe several procedures from the BIAS and PROFIL packages, which are used throughout the program. Along with the procedure name we give the data type it returns and a short description.

6.2 Procedures for computation of the left side of the Lorenz equation, its first and second derivatives

The procedures below are used for the computation of $\mathbf{f}(\mathbf{y})$, $\mathbf{f}'(\mathbf{y})\cdot\mathbf{h}$, $\mathbf{f}''(\mathbf{y})\cdot(\mathbf{h}_1, \mathbf{h}_2)$ and $\mathbf{f}''(\mathbf{y}) \cdot (\mathbf{h}, \mathbf{h})$.

```
INTERVAL_VECTOR LeftSide( INTERVAL_VECTOR & Y ){
  /* returns Left side of Lorenz equation */
  INTERVAL_VECTOR       LS(3);
  LS( 1 ) = S * ( Y( 2 ) - Y( 1 ) );
  LS( 2 ) = R * Y( 1 ) - Y( 2 ) - Y( 1 ) * Y( 3 );
  LS( 3 ) = Y( 1 ) * Y( 2 ) - Q * Y( 3 );
  return( LS );
}

INTERVAL_VECTOR FPrim( INTERVAL_VECTOR & Y, INTERVAL_VECTOR & H ){
  INTERVAL_VECTOR       FP(3);
  /* returns first derivative of the left side of Lorenz system
     at point Y on vector H */
  FP( 1 ) = -S      *H(1) + S   *H(2) ;
  FP( 2 ) = (R-Y(3))*H(1) - 1 *H(2) - Y(1)*H(3);
  FP( 3 ) = Y(2)    *H(1) + Y(1)*H(2) - Q   *H(3);
  return( FP );
}

INTERVAL_VECTOR FBis( INTERVAL_VECTOR & H1, INTERVAL_VECTOR & H2 ){
  INTERVAL_VECTOR       FP(3);
  /* returns second derivative of the left side of Lorenz system
     on vectors H1, H2 (F''(Y) does not depend on Y) */
  FP( 1 ) = 0 ;
  FP( 2 ) = - H1( 1 ) * H2( 3 ) - H1( 3 ) * H2( 1 ) ;
  FP( 3 ) =   H1( 1 ) * H2( 2 ) + H1( 2 ) * H2( 1 ) ;
  return( FP );
}

INTERVAL_VECTOR FBis2( INTERVAL_VECTOR & H ){
  INTERVAL_VECTOR       FP(3);
  /* returns second derivative of the left side of Lorenz system
     on vectors H1=H2=H (F''(Y) does not depend on Y) */
  FP( 1 ) = 0 ;
  FP( 2 ) = - 2 * H( 1 ) * H( 3 ) ;
  FP( 3 ) =   2 * H( 1 ) * H( 2 ) ;
  return( FP );
}
```

The following procedure computes interval vector `PT` containing trajectory starting from interval vector `P`, after time $[0, \mathtt{h}]$.

```
BOOL GetTraj( INTERVAL_VECTOR P, INTERVAL_VECTOR & PT,
              INTERVAL h ){
  INTERVAL_VECTOR       EpsVectVar( 3 );
```

```
  INTERVAL              hfull,h2full2;
  INTERVAL_VECTOR       LS( 3 ),Y( 3 ),YEps( 3 );

  /* EpsVect is an 3D INTERVAL_VECTOR defined as:
     EpsVect = 0.005 * [-1,1]x[-1,1]x[-1,1] */
  EpsVectVar = EpsVect;
  hfull = Hull( 0,h );
  h2full2 = Sqr( Hull( 0,h ) ) / 2;
  LS = LeftSide( P );
  while (TRUE) {
    Y = P + hfull * LS + EpsVectVar ;
    /* make YEps a little bigger than Y */
    YEps = Succ( Y ) ;
    PT = P + hfull * LS + h2full2 * FPrim( YEps,LeftSide( YEps ) );
    /* PT < Y returns TRUE if every component of PT is contained
       in the interior of the corresponding component of Y:
       PT subset int(Y) */
    if ( PT < Y ) return( TRUE );
    EpsVectVar *= 2;
    if ( Max( Sup( EpsVectVar ) ) > 1000 ) return( FALSE );
  }
}
```

The following procedure increases `Epsilon` by the radius of the interval vector `P` while `P` is shrinked to the point. New values of `P` and `Epsilon` are chosen in such a way that $\mathrm{B}(\texttt{POld}, \texttt{EpsilonOld}) \subset \mathrm{B}(\texttt{P}, \texttt{Epsilon})$ and `P` is a point.

```
void DecPIncEpsilon( INTERVAL_VECTOR & P, REAL & Epsilon ){
  INTERVAL              Eps;
  INTERVAL_VECTOR       PMid( 3 );
  /* P is shrinked to the point and Epsilon increased in such
     a way that B(POld,EpsilonOld) \subset B(P,Epsilon) */

  /* Mid( P ) returns the midpoint of the interval vector P */
  PMid = Mid( P );
  /* Norm( IV ) computes the 2-norm of the INTERVAL_VECTOR IV*/
  Eps = Norm(P - PMid);
  Epsilon = Sup(Epsilon + Eps);
  P = PMid;
}
```

6.3 Procedure for computation of one integration step

```
BOOL NextPointExactTaylor4( INTERVAL_VECTOR & P0, REAL & Epsilon0,
```

```
                        INTERVAL_VECTOR & P2, REAL & Epsilon2 ){
INTERVAL_VECTOR       Y( 3 ),Err( 3 );
INTERVAL_VECTOR       P1( 3 );
INTERVAL              TempInt,F3;
BOOL                  Done;
REAL                  L,Epsilon1;
INTERVAL_VECTOR       F( 3 ),FPF( 3 ),FBFF( 3 ),FPFPF( 3 );

P1 = P0;
Epsilon1 = Epsilon0;

/* OneVect is a 3D INTERVAL_VECTOR defined as
   OneVect = [-1,1]x[-1,1]x[-1,1] */
/* The Euclidean ball B(P1,Epsilon1) is a subset of
   P1 + Epsilon * OneVect */

/* Computation of Y - interval vector containing the trajectory
   starting from set P1 + Epsilon1 * OneVect after time [0,h] */
if (!GetTraj(P1 + Epsilon1 * OneVect,Y,h)) return( FALSE );

/* Computation of logarithmic norm on this set */
if (!LogNorm( Y,L )) return( FALSE );

/* Transversality condition --- we check if vector field is
   transversal to the Poincare plane on the set Y */
/* procedure Intersection( A,B,C ) returns TRUE if B and C has
   nonempty intersection */
if ( Intersection( TempInt,Y( 3 ),PoincValue )){
  /* Y( 3 ) and PoincValue has nonempty intersection */
  /* LeftSide3( Y ) returns the third component of the
     left side of Lorenz equations: z'( Y ) = LeftSide3( Y ) */
  F3 = LeftSide3( Y ) ;
  /* if vector field is not transversal i.e., 0 is in F3
     then return FALSE */
  if ( 0 <= F3 ) {
    printf( "transversality error" ) ;
    return( FALSE );
  }
}

/* Increasing of the radius of the ball
   (according to logarithmic norm) */
Epsilon2 = Sup( Exp( Hull( L ) * h ) * Epsilon1 );

/* fourth-order TAYLOR method */
```

```
  F = LeftSide( P1 );
  FPF = FPrim( P1,F );             /* FPF   = F'F */
  FPFPF = FPrim( P1,FPF );         /* FPFPF = F'F'F */
  FBFF = FBis2( F );               /* FBFF  = F''FF */
  P2 = P1 + h * F + h2*FPF / 2 + h3*( FBFF + FPFPF ) / 6 +
       h4*( 3 * FBis( FPF,F ) + FPrim( P1,FBFF ) +
            FPrim( P1,FPFPF ) )/24;
  /* end of TAYLOR method */

  /* Taylor series truncation error */
  /* Error = (h^5/120)*(4*f''ff''ff+4*f''f'f'ff+3*f''f'ff'f+
                        3*f'f''ff'f+f'f'f''ff+f'f'f'f'f)*/
  /* This error is computed on Y containing trajectory
     starting from set P1 after time [0,h] */
  if (!GetTraj( P1,Y,h ) ) return( FALSE );
  F = LeftSide( Y );
  FPF = FPrim( Y,F );               /* FPF   = F'F */
  FPFPF = FPrim( Y,FPF );           /* FPFPF = F'F'F */
  FBFF = FBis2( F );                /* FBFF  = F''FF */
  Err = ( 4 * FBis( F,FBFF ) + 4 * FBis( FPFPF,F ) +
          3 * FBis2( FPF ) + 3 * FPrim( Y,FBis( F,FPF ) ) +
          FPrim( Y,FPrim( Y,FBFF ) ) +
          FPrim( Y,FPrim( Y,FPFPF ) ) ) * h5 / 120;
  /* Error addition */
  P2 = P2 + Err ;
  return( TRUE );
}
```

6.4 Procedure for the Poincaré map

The procedure `PoincFun` finds a 2D interval vector `EnP` such that $\mathbf{P}(\texttt{StP}) \subset \texttt{EnP}$.

```
BOOL PoincFun(INTERVAL_VECTOR & StP, INTERVAL_VECTOR & EnP,
              INT SectionType){
  /* SectionType - Type of computation of the section */
  /* SectionOpt - with optimalization */
  /* SectionNoComp - without computation image - just checking if
                     Poincare map is defined */

  #define               LogNorm
  /* LogNorm defined - size of neighborhood is increased
     according to Logarithmic norm, region is the ball
     B(P,Epsilon), where P is the point INTERVAL_VECTOR
     -- better results */
```

```
/* LogNorm undefined - region is the INTERVAL_VECTOR P
   -- worse results */

INTERVAL_VECTOR  P( 3 ),P2( 3 ),PPP( 3 ),PPoincFull( 3 ),Y( 3 );
REAL             Epsilon,Epsilon2;
BOOL             FirstSection,SecondSecStart;
BOOL             done,OK;

P2( 1 ) = StP( 1 );
P2( 2 ) = StP( 2 );
P2( 3 ) = PoincValue;
Epsilon2 = 0;

/* initiate values h=TimeStep,h2=h^2,...,h5=h^5  */
h = TimeStep;
h2 = Sqr( h );   h3 = h * h2;
h4 = h2 * h2;   h5 = h * h4;

done = FALSE;
OK = TRUE;
FirstSection = FALSE;
SecondSecStart = FALSE;

while ( !done ) {
  P = P2;
  Epsilon = Epsilon2;

  #ifdef LogNorm
    /* Increasing Epsilon and shrinking P */
    DecPIncEpsilon( P,Epsilon );
  #endif
  /* next integration step */
  if (!NextPointExactTaylor4( P,Epsilon,P2,Epsilon2 )) {
    printf( "NextPoint Error\n" );
    done = TRUE;
    OK = FALSE;
  }
  #ifdef LogNorm
    /* Increasing Epsilon2 and shrinking P2 to the point */
    DecPIncEpsilon( P2,Epsilon2 );
  #endif

  if ( ( Epsilon2 > 200 ) ) {
    printf( "Epsilon error\n" );
    done = TRUE;
```

```
      OK = FALSE;
    }

    /* searching for first section */
    if ((!FirstSection) &&
        (Inf( P2( 3 ) - Epsilon2 ) > Sup( PoincValue ) ) ) {
      /* End of the first section detected */
      FirstSection = TRUE;
    }

    /* searching for the start of second section */
    if ( !SecondSecStart && FirstSection &&
         (Inf( P2( 3 ) - Epsilon2 ) < Sup( PoincValue ) ) ) {
      /* Start of the second section detected  */
      SecondSecStart = TRUE;
      PPoincFull = P;
    }
    /* searching for the end of second section */
    if ( SecondSecStart &&
       (Sup( P2( 3 ) + Epsilon) < Inf( PoincValue ) ) ) {
      /* condition in the if above means that the Second
         section start has already been detected and
         the ball B(P2,Epsilon) lies after section */
      /* End of second section detected */
      done = TRUE;
    }

    /* section computation */
    if ((SectionType==SectionOpt) && SecondSecStart) {
      if (!GetTraj(P + Epsilon * OneVect,PPP,h)) {
        /* phi(B(P,Epsilon),[0,h]) subset PPP */
        OK = FALSE;
        done = TRUE;
        printf( "SecondSection Error\n" );
      }
      /* incr. PPoincFull to include phi(B(P,Epsilon),[0,h]) */
      PPoincFull = Hull( PPoincFull,PPP );
    }
  }
  EnP( 1 ) = PPoincFull( 1 );
  EnP( 2 ) = PPoincFull( 2 );

  if (!OK) printf( "PoincFun Error\n" );
  return( OK );
}
```

References

[ABS1] V.S. Afraimovich, V.V. Bykov and L. P. Shil'nikov *On the appearance and structure of Lorenz attractor,* DAN SSSR, 234, pp. 336–339, 1977.

[ABS2] V.S. Afraimovich, V.V. Bykov and L. P. Shil'nikov *On the structurally unstable attracting limit set of Lorenz type attractors,* Tran. Mosc. Soc. 2, pp. 153–215, 1983.

[Ch] X. Chen, *Lorenz Equations, Part III: Existence of Hyperbolic Sets*, preprint 1995.

[Co] C.Conley,*Isolated Invariant Sets and the Morse Index*, CBMS Regional Conf. Ser. Math., no. 38, AMS, Providence, R.I., 1978.

[E75] R. Easton, *Isolating blocks and symbolic dynamics* , J. Diff. Eqs., 17, pp. 96–118, 1975.

[E89] R. Easton, *Isolating blocks and epsilon chains for maps*, Physica D, 39, pp. 95–110, 1989.

[G] Z. Galias, *Positive topological entropy of Chua's circuit: a computer assisted proof*, Int. J. of Bifurcation and Chaos, vol. 7, no. 2, pp. 331–349, 1997.

[GH] J. Guckenheimer and P. Holmes *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields*, Springer–Verlag: New York, Berlin, Heidelberg, Tokyo, 1983.

[Gr] M. J. Greenberg, *Lectures on Algebraic Topology*, W.A. Benjamin Inc., 1973.

[HHTZ] B. Hassard, S. Hastings, W. Troy, and J. Zhang, *A computer proof that the Lorenz equations have "chaotic" solutions*, Appl. Math. Letters, 7, pp. 79–83, 1994.

[HT] S.P. Hastings and W.C. Troy, *A shooting approach to the Lorenz equations*, Bulletin (New Series) of the American Mathematical Society, 27, pp. 298-303, 1992.

[HNW] E. Hairer, S.P. Nørsett, G. Wanner, *Solving ordinary differential equations. I. Nonstiff problems*, Springer–Verlag.

[K] O. Knüppel, *PROFIL — programmer's runtime optimized fast interval library*, Technical University Hamburg-Harburg, Bericht 93.4, July 1993.

[L] E. Lorenz, *Deterministic non-periodic flow*, J. Atmos. Sci 20, 130, 1963.

[Lo] R.J. Lohner, *Computation of Guaranteed Enclosures for the Solutions of Ordinary Initial and Boundary Value Problems*, in: Computational Ordinary Differential Equations, J.R. Cash, I. Gladwell Eds., Clarendon Press, Oxford, 1992.

[Mi] K. Mischaikow, *The Conley index theory: some recent developments*, CIME Lectures, preprint.

[MM1] K. Mischaikow and M. Mrozek, *Chaos in the Lorenz Equations: a Computer Assisted Proof*, Bull. of AMS, vol. 32, pp. 66–72, 1995.

[MM2] K. Mischaikow and M. Mrozek, *Isolating neighborhoods and chaos*, Jap. J. Ind. & Appl. Math., 12, pp. 205–236, 1995.

[MM3] K. Mischaikow and M. Mrozek, *Chaos in the Lorenz Equations: A Computer Assisted Proof. Part II: Details* , preprint CDSNS95-222, to appear in Mathematics and Computation.

[MMS] K. Mischaikow, M. Mrozek and A. Szymczak, *Chaos in the Lorenz Equations: A Computer Assisted Proof. Part III:The classical case* , in preparation.

[Mo] J. Moser, *Stable and Random Motions in Dynamical Systems*, Princeton Univ. Press, 1973.

[M] M. Mrozek, *Leray Functor and Cohomological Conley for Discrete Dynamical Systems*, Trans. Amer. Math. Soc., vol. 318, pp. 149–178, 1990.

[S67] S. Smale, *Differentiable Dynamical Systems*, Bull. Amer. Math. Soc., vol. 73, pp. 747–817, 1967.

[Sp] C. Sparrow, *The Lorenz Equations.* Springer–Verlag: New York, Heidelberg, Berlin, 1982.

[SA] H. Spreuer, E. Adams, *On the strange attractor and Transverse Homoclinic Orbits for Lorenz equations*, J. Math. Anal. and Appl., 190, 1995, pp. 329–360.

[Sh] L.P. Shil'nikov, *Chua's circuit: rigorous results and future problems* , Int. J. of Bifurcations and Chaos, vol. 4, no. 3, pp. 489–519, 1994.

[Sz] A. Szymczak, *A Combinatorial Procedure for Finding Isolating Neighborhoods and Index Pairs* , Proc. Royal Soc. Edinburgh., to appear.

[W] R.J. Williams *The structure of Lorenz attractor*, Lect. Notes in Math., 615, pp. 94–112, Springer-Verlag, Berlin, 1997.

[Z96a] P. Zgliczyński, *Fixed point index for iterations of maps, topological horseshoe and chaos*, Topological Methods in Nonlinear Analysis, 1996, vol. 8, no. 1, pp. 169–177.

[Z96b] P. Zgliczyński, *Rigorous verification of chaos in the Rössler equations*, Proc. of SCAN-95, 287–292, Math. Research, vol. 90, Akademie Verlag, 1996

[Z97a] P. Zgliczyński, *Computer assisted proof of chaos in the Rössler equations and the Hénon map*, Nonlinearity, vol. 10, no. 1, pp. 243–252, 1997.

[Z97b] P. Zgliczyński, *A Computer assisted proof of the horseshoe dynamics in the Hénon map*, accepted for publication in Random & Computational Dynamics.