Invited Paper

Computer assisted proof of chaos in the Muthuswamy-Chua memristor circuit

Zbigniew Galias^{1 a)}

¹ AGH University of Science and Technology
Department of Electrical Engineering
al. Mickiewicza 30, 30-059 Kraków, Poland

^{a)} galias@agh.edu.pl

Received January 9, 2014; Published July 1, 2014

Abstract: It is shown that the Muthuswamy Chua memristor circuit is chaotic in the topological sense. A rigorous analysis of the circuit is performed using interval analysis based tools.

Key Words: memristor, chaos, interval arithmetic

1. Introduction

Memristor is a circuit element characterized by a nonlinear relation between charge and flux. Its resistance depends on the history of current flowing through the element or the voltage across the element. Memristors were theoretically introduced in 1971 by prof. L.O. Chua [3]. Memristive behaviour was recognized in a nano-scale double-layer TiO₂ film by researchers at Hewlett-Packard Labs in 2008 [20]. Afterwards, memristors have received significant attention due to a number of possible applications including large capacity nonvolatile memories and neuromorphic systems.

Formally, a *current controlled time-invariant memristor* is defined by the following set of equations (a voltage-controlled memristor may be defined in a similar manner):

$$v = R(z, i)i, \quad \frac{dz(t)}{dt} = f(z, i),$$

where $z \in \mathbb{R}^n$ is the vector of state variables, which do not depend on any external voltages and currents. The function $R(z, i)$ is called the *memristance*. In a special case when there is a single state variable, which is the charge, i.e. $z(t) = q(t)$, the above set of equations defines an *ideal charge-controlled memristor*. For the ideal memristor there exist a constitutive relation between the charge q and the magnetic flux φ in the form $\varphi = f_M(q)$, and the memristance $R(q) = \frac{df_M(q)}{dq}$ is a function of the charge q . In this context the memristor can be considered as the fourth basic passive circuit element (apart from the resistor, inductor, and capacitor) [4].

Analysis of dynamical phenomena in memristive systems attracted a lot of attention since the recognition of memristive behaviour in nano-scale [20]. In [10], the authors analysed systems with memristors defined by the piecewise-linear charge-flux relationship. Various nonlinear phenomena were obtained in Chua's oscillator by replacing Chua's diodes with memristors. In [15], several chaotic oscillators obtained by using memristors as the nonlinear devices in Chua's circuit family

were proposed. The main conclusion from this research is that memristors are useful for designing nonlinear oscillators. The existence of chaotic dynamics have been observed in simulations of various electronic oscillators containing memristors [7, 11, 12, 22]. Certain bifurcation phenomena existing in nonlinear oscillators based on mem-devices are reported in [18].

All these results are mainly based on simulations and sometimes computing certain characteristics of nonlinear systems like Lyapunov exponents and simulation based bifurcation analysis. It is however well known that due to rounding errors results obtained by numerical simulations of nonlinear systems may be unreliable. Using finite representations of real numbers causes that results found by numerical simulations are not exact, except in some special cases. Rounding errors accumulate and are propagated in further computations. For more complex problems this may result in totally wrong answers. Several examples are shown in [8]. The problem of error propagation is especially important when we compute trajectories of chaotic systems since errors introduced in each computation step grow exponentially due to sensitivity to initial conditions.

In this work, we extend the analysis of nonlinear behaviours in memristive systems using rigorous computational tools. Dynamical phenomena studied in this work include the existence of complex trapping regions and the existence of complex symbolic dynamics. To carry out rigorous computations we use interval analysis [2, 14, 17].

The layout of the paper is following. In Section 2, the memristor circuit to be studied is defined. In Section 3, we define the return map and describe how to evaluate it in a rigorous way. The existence of a trapping region for the return map is proved in Section 4. Several computer-assisted proof techniques are compared in terms of computation time. In Section 5 the existence of complex symbolic dynamics is established, and a positive lower bound for the topological entropy of the return map is found, thus proving that that system is chaotic in the topological sense.

2. Third order memristor circuit

Let us consider a three element memristor circuit [16] shown in Fig. 1(a). It consist of two linear elements (the capacitor C and the inductor L) and a nonlinear memristor defined by the following equations:

$$\begin{aligned} u_M &= \beta(z^2 - 1)i_M, \\ \frac{dz}{dt} &= i_M - \alpha z - i_M z, \end{aligned}$$

where u_M is the voltage across the device, i_M is the current flowing through the device, and z is the internal variable of the memristor device.

Dynamics of the circuit is described by

Fig. 1. (a) The Muthuswamy-Chua memristor circuit, (b) an example trajectory of the circuit.

$$\begin{aligned}
\frac{dx}{dt} &= \frac{y}{C}, \\
\frac{dy}{dt} &= -\frac{1}{L} (x + \beta(z^2 - 1)y), \\
\frac{dz}{dt} &= -y - \alpha z + yz,
\end{aligned} \tag{1}$$

where $x(t) = u_C(t)$ is the voltage across the capacitor C , $y(t) = i_L(t)$ is the current through the inductor L and z is the internal state of the memristor. We will consider the circuit with the following parameter values: $C = 1$, $L = 3$, $\beta = 1.5$, $\alpha = 0.6$. A trajectory of the system of length $T = 1000$ is shown in Fig. 1(b). It was obtained by numerical integration of (1) using the fourth-order Runge-Kutta method and the time step $\tau = 0.1$.

3. Return map and its rigorous evaluation

Let us choose the return map P defined by the plane $\Sigma = \{(x, y, z) \in \mathbb{R}^3 : x = 0\}$. Formally, the *return map* $P: \Sigma \mapsto \Sigma$ is defined as

$$P(x) = \varphi(\tau(x), x), \tag{2}$$

where $\varphi(t, x)$ is the trajectory of the system based at x , and $\tau(x)$ is the time needed for the trajectory $\varphi(t, x)$ to reach Σ . The return map P is sometimes referred to as the Poincaré map.

A trajectory of P composed of 10000 points is shown in Fig. 2. Intersections of the trajectory of the continuous system with the plane $x = 0$ in two different directions ($\dot{x} < 0$ and $\dot{x} > 0$) are plotted using different colors.

Fig. 2. A trajectory of the return map P .

The map P can be evaluated using rigorous integration methods based on interval analysis [2, 14, 17]. In interval analysis intervals are used instead of real numbers. The closed interval with endpoints $a \leq \bar{a}$ is denoted by $\mathbf{a} = [\underline{a}, \bar{a}] = \{a : \underline{a} \leq a \leq \bar{a}\}$. An interval vector is a Cartesian product of m intervals $\mathbf{v} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m) = \{(a_1, a_2, \dots, a_m) : a_i \in \mathbf{a}_i \text{ for } i = 1, \dots, m\}$. It corresponds to a rectangle in $\mathbb{R}^2$, and a box in $\mathbb{R}^m$. We will use bold letters to denote intervals, interval vectors and matrices, and usual math italic to denote real quantities. On the set of intervals basic arithmetic operators $(+, -, \times, \div)$ are defined in such a way that the result of an operation on intervals is the smallest interval containing results of the corresponding real operation for all combinations of values from these intervals. The result of arithmetic operations on intervals can be found by performing “real” arithmetic operations on their endpoints. In practice, it is impossible to perform arithmetic operations (real or interval) with infinite precision. We are limited by representations of finite precision. When interval arithmetic is implemented on a computer one has to make sure that the result of an operation on intervals contains the results of the corresponding real operation for all combinations of values from these intervals. In the best implementation of interval arithmetic the left (right) endpoint found by a computer is the largest (smallest) representable real number not larger (not smaller) than the correct value. This is obtained by using directed rounding when performing “real” computations on interval endpoints.

For the evaluation of P we use the CAPD library [1]. For the integration of the vector field (1) the rigorous Taylor integration method of order 30 with automatic control of the time step is used. The Lohner method is used to reduce the wrapping effect [13]. The evaluation of the return map by means of interval arithmetic is rigorous in the sense that the result found by a computer encloses the true result, i.e.

$$P(\mathbf{x}) = \{P(x) : x \in \mathbf{x}\} \subset \mathbf{y}, \quad (3)$$

where $\mathbf{x}$ is the interval vector for which the return map is evaluated and $\mathbf{y}$ is the enclosure found by a computer.

The full implementation of the procedure for the evaluation of the return map P is presented in Algorithm 1. The first line defines variable names, parameter names, and the vector field. Next four lines define parameter values. Note that since the number 0.6 is not machine representable it cannot be initialized as `VectorField.setParameter("a",0.6)`. In the next three lines the Taylor solver of order 30, the section $x = 0$ and the return map are defined. The line `C0Rect2Set set(x);` defines the Lohner method representation of the set to be used during the integration, and the last line calls the procedure to do actual computations, i.e. find an enclosure of the image of the input interval vector $\mathbf{x}$ under P . On exit $\mathbf{y}$ is an enclosure of $P(\mathbf{x})$ and $\mathbf{t}$ is an enclosure of the return time.

Algorithm 1 Procedure for the evaluation of the return map

```
void ReturnMap(IVector x, interval & t, IVector & y){
    IMap VectorField("par:L,C,a,b;var:x,y,z;fun:y/C,-(x+b*(z^2-1)*y)/L,-y-a*z+y*z;");
    VectorField.setParameter("C",1.0);           // C=1.0
    VectorField.setParameter("L",3.0);           // L=3.0
    VectorField.setParameter("a",interval(3.)/5.); // b=0.6
    VectorField.setParameter("b",1.5);           // b=1.5
    ITaylor ISolver(VectorField,30);             // Taylor method of order 30
    ICoordinateSection ISection(3,0);            // the section is x=0
    IPoincareMap IPM(ISolver,ISection,poincare::Both); // return map
    t=0.0;
    C0Rect2Set set(x);
    y=IPM(set,t);
}
```

Let us note that computing rigorous enclosures of return maps using the CAPD library is very easy. Reducing the wrapping effect by means of the Lohner method is done internally and is hidden from the user. Test of the Algorithm 1 and its output are shown as Algorithm 2.

The enclosure $\mathbf{y}$ found by the algorithm is shown as the red rectangle in Fig. 3(a). For comparison, the border of the set $P(\mathbf{x})$ is shown using blue color. $P(\mathbf{x})$ has a shape of a parallelogram and can be enclosed in the interval vector with diameter 0.0014×0.00057 . The diameter of the enclosure $\mathbf{y}$ is 0.00257×0.00195 . Using the Lohner algorithm reduces the wrapping effect and in consequence the overestimation is limited. Without using the Lohner method the procedure fails to evaluate $P(\mathbf{x})$.

Another example is shown in Fig. 3(b). In this example the true solution is a very narrow parallelogram shaped area. The overestimation is even smaller than in the first example.

4. The existence of a positively invariant set for the return map

In this section we will show that there exist a positively invariant set for the return map P . We say that a set A is *positively invariant* (or a *trapping region*) for the map P if $P(x) \in A$ for all $x \in A$. Establishing the existence of a trapping region is an important step in the analysis of the dynamics of a nonlinear dynamical system. A trajectory starting in a positively invariant set is trapped within this set. It follows that the trapping regions contains an attractor and its accurate enclosure can be found by iterating the map over the trapping region.

Algorithm 2 Test for the ReturnMap procedure

```
void TestReturnMap(){
    interval t;
    IVector x(3),y;
    x[0] = 0.0;
    x[1] = interval(0.277,0.278);
    x[2] = interval(-1.085,-1.084);
    ReturnMap(x,t,y);
    cout << "Return time: " << t << endl;
    cout << "P(x): " << y << endl;
}

//output
//Return time: [5.05476370833807, 5.05789781673181]
//P(x): {[ -0.00149675286210518, 0.00149628025121867],
//[-0.478943153519594, -0.476373461426272], [0.18627510663872, 0.188219911113476]}
```

Fig. 3. Examples of finding enclosures of $P(\mathbf{x})$, (a) $\mathbf{x} = (0, [0.277, 0.278], [-1.085, -1.084])$. (b) $\mathbf{x} = (0, [-0.478, -0.477], [0.187, 0.188])$.

Limit sets like equilibria and periodic orbits are simple examples of positively invariant sets. Here, we are interesting in finding a trapping region of positive volume, and in particular a trapping region enclosing the numerically observed attractors.

The existence of a positively invariant set is done in two steps. First we construct a candidate set A , and then we prove that $P(A) \subset A$.

The candidate set Ω is shown in Fig. 4(a). It is composed of two polygons with 11 and 13 edges, respectively. One can see that the candidate set encloses a trajectory of P . The image of a number of points belonging to the border of Ω is shown in Fig. 4(b). One can see that test points are mapped to the interior of Ω , and hence Ω is a good candidate for a trapping region. For now, the evaluation of P is performed nonrigorously.

In order to prove that Ω is indeed a trapping region we use the generalized bisection procedure. To this end, the set Ω is first enclosed in a set of interval vectors (rectangles) in such a way that each rectangle contains one component of Ω . These rectangles are added to the list of rectangles to be processed. In the following steps interval vectors from the list are processed. We take the rectangle $\mathbf{x}$ from the list, compute enclosure $\mathbf{y}$ of $P(\mathbf{x})$ and verify whether $\mathbf{y} \subset \Omega$. If the condition $\mathbf{y} \subset \Omega$ is not fulfilled the rectangle $\mathbf{x}$ is split into four rectangles. Rectangles, which have nonempty intersection with Ω are added to the list. After $\mathbf{x}$ has been processed, we proceed with the next element from the list. When the list is empty the proof that $P(\Omega) \subset \Omega$ is completed.

The generalized bisection procedure has been applied to the candidate set Ω . In the course of the procedure a covering composed of 13581 interval vectors has been generated. The covering is shown in Fig 5(a). One can see that the covering rectangles are not of uniform size. They are smaller in regions

Fig. 4. (a) Trapping region candidate Ω encloses the numerically observed attractor, (b) images of test points belonging to the border of Ω computed nonrigorously.

Fig. 5. (a) Covering of the trapping region Ω by 13581 rectangles generated by the generalized bisection procedure, (b) image of the covering is enclosed in the trapping region.

where fine splitting is necessary. Enclosures of images of covering rectangles are shown in Fig. 5(b). It has been verified that all images are enclosed in Ω and hence it has been proved that Ω is positively invariant. During the procedure the total number of 18338 rectangles have been processed. For 4757 rectangles the enclosure condition has not been satisfied and these rectangles were split into smaller parts. The computer assisted proof took 153 seconds to complete on a single core 3.1 GHz processor.

It is possible to speed up computations by using the fact that the return map is injective (one-to-one) and continuous, which is a consequence of the property of uniqueness of solutions of the dynamical system. For injective and continuous maps the interior of a set is mapped to the interior of the image of the border. Hence, it is sufficient to show that the image of the border $\partial\Omega$ is enclosed in Ω and that the map is well defined on Ω . Both conditions can be verified using the generalized bisection method described above with the difference that for the existence proof we only find enclosure $\mathbf{y}$ of $P(\mathbf{x})$ without verifying that $\mathbf{y} \subset \Omega$. Successful evaluation of P over $\mathbf{x}$ ensures that P is well defined on $\mathbf{x}$.

Covering of $\partial\Omega$ by interval vectors generated by the generalized bisection procedure for the proof that $P(\partial\Omega) \subset \Omega$ is shown in Fig. 6(a). The covering is composed of 405 boxes. The proof that $P(\partial\Omega) \subset \Omega$ took 4 seconds. Covering of Ω by 4294 rectangles for the proof that P is well defined on Ω is shown in Fig. 6(b). Note that the covering rectangles are larger than in Fig. 5(b), and in consequence fewer of them are necessary to cover Ω . The proof that P is well defined on Ω took 77 seconds, and hence the total time necessary to complete the proof that Ω is a trapping region was reduced almost two times to 81 seconds.

Fig. 6. (a) Covering $\partial\Omega$ by 405 rectangles for the proof that $P(\partial\Omega) \subset \Omega$, (b) Covering of Ω by 4294 rectangles for the proof that P is well defined in Ω .

Let us now compare how the choice of the return map influences computation performance. Let us consider the return map P_1 defined by the section $\Sigma_1 = \{(x, y, z) \in \mathbb{R}^3 : x = 0, \dot{x} < 0\}$, i.e. we count only those intersections with Σ defining the map P for which $\dot{x} < 0$ at the intersection point. Let us choose the polygon Ω_1 lying in the left part of Fig. 4(a) as a candidate of the trapping region. For $x \in \Omega_1$ we have $P_1(x) = P(P(x))$, i.e. two iterations of P are equivalent to one iteration of P_1 . Time needed to prove that $P_1(\Omega_1) \subset \Omega_1$ is 496 seconds, which is three times more than time needed to prove that $P(\Sigma) \subset \Sigma$.

It is interesting to note that if we choose the return map P_2 defined by the section $\Sigma_1 = \{(x, y, z) \in \mathbb{R}^3 : x = 0, \dot{x} > 0\}$ and the polygon Ω_2 lying in the right half of Fig. 4(a) as a candidate of the trapping region, then the time needed to prove that $P_2(\Omega_2) \subset \Omega_2$ exceeds half an hour.

Table I. Details of computer assisted proofs.

Statement	Covering Size	Proof time [s]	Comments
$P(\Omega) \subset \Omega$	13581	153	Fig. 5
$P(\Omega) \subset \Omega$	$405 + 4294 = 4699$	$4 + 77 = 81$	handling border and interior separately, Fig. 6
$P_1(\Omega_1) \subset \Omega_1$	41701	496	
$P_2(\Omega_2) \subset \Omega_2$	139930	2318	

Covering sizes and computation time for the four computer-assisted proofs presented above are collected in Table I. From the discussion presented above, it follows that introducing intermediate sections or as in our case allowing intersection with both directions may shorten the total computation time. This is a consequence of the fact that intermediate sections reduce return times for the return map, and in consequence the wrapping effect and overestimation of enclosures are significantly smaller.

5. The existence of complex symbolic dynamics and positive topological entropy

In this section, we prove the existence of complex symbolic dynamics, i.e. symbolic dynamics with positive topological entropy, for the map P . For that we will use the method of covering relations. Let us briefly recall the definition and properties of covering relations [9, 21]. We will limit ourselves to the two-dimensional case. Let us consider a continuous map $f: \mathbb{R}^2 \mapsto \mathbb{R}^2$. Let us select p quadrangles $Q_1, Q_2, \dots, Q_p$. For each quadrangle we choose two opposite edges and call them “horizontal”. Two other edges are called “vertical”. We say that Q_i f -covers Q_j if (i) the image of Q_i under f has empty intersection with the horizontal edges of Q_j and (ii) the images of vertical edges of Q_i have empty intersection with Q_j and are located geometrically on the opposite sides of Q_j (i.e. $f(Q_i)$ can be continuously deformed to the set enclosed in the stripe defined by the horizontal edges of Q_j without any intersection with the edges of Q_j and then images of vertical edges of Q_i lie in the stripe on the

opposite sides of N_j . In the following, instead of quadrangles we will use topological quadrangles, i.e. sets which can be continuously deformed to quadrangles.

In the example shown in Fig. 7 the quadrangle Q_1 f -covers Q_1 and Q_2 , while the quadrangle Q_2 f -covers Q_1 only.

Fig. 7. Covering relation examples. Vertical edges and their images are plotted using thick lines. Q_1 f -covers Q_1 and Q_2 , Q_2 f -covers Q_1 .

Let us now describe how to use covering relations involving a given map f to obtain estimates for the topological entropy of this map. Let us assume that we have sets $Q_1, Q_2, \dots, Q_p$ and some covering relations between them. Let us define the transition matrix $A = (a_{ij})_{i,j=1}^p$

$$a_{ij} = \begin{cases} 1 & \text{if } Q_i \text{ } f\text{-covers } Q_j, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The existence of covering relations described by the transition matrix A means that f is semiconjugate with a subshift of finite type with the transition matrix A . From this fact one can obtain estimates of the topological entropy of the map f . The following theorem states the relation between the transition matrix and topological entropy of f [9, 19].

Theorem 1 Let us assume that the quadrangles $Q_1, Q_2, \dots, Q_p$ are pairwise disjoint, and that the matrix $A \in \mathbb{R}^{p \times p}$ describes covering relations involving map f between the quadrangles Q_i . Then the topological entropy of f is not less than the logarithm of the leading eigenvalue of A .

Figure 8(a) shows candidates for the quadrangles $Q_1, Q_2, \dots, Q_8$ supporting complex symbolic dynamics, an example trajectory of P and the trapping region Ω . Vertical edges of quadrangles Q_i are plotted using black color. One can see that horizontal edges of quadrangles have empty intersection with the trajectory.

Figure 8(b) shows images of quadrangles Q_i computed nonrigorously. Vertical edges and their images are plotted using black color. One can see that Q_1 P -covers Q_7 , Q_2 P -covers Q_5 , Q_3 P -covers Q_7 and Q_8 , Q_4 P -covers Q_6 , Q_5 P -covers Q_1 , Q_6 P -covers Q_2 , Q_7 P -covers Q_3 , and Q_8 P -covers Q_4 . These coverings can be presented in the form of the following transition matrix:

Fig. 8. (a) Candidates for quadrangles Q_i supporting complex symbolic dynamics, (b) images of quadrangles Q_i computed nonrigorously.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (5)$$

Below, we show details of a computer assisted proof that indeed the nine covering relations corresponding to non-zero entries in the transition matrix (5) exist. The existence of each covering relation is proved independently. Recall that to prove that Q_i P -covers Q_j we have to show that (i) the image of Q_i under f is enclosed in the topological stripe defined by the horizontal edges of Q_j and that (ii) the images of vertical edges of Q_i have empty intersection with Q_j and are located geometrically on the opposite sides of Q_j . Since the quadrangles Q_i are enclosed in the trapping region Ω (compare Fig. 8(a)), it follows that the map P is well defined and injective on $\bigcup_i Q_i$. Hence, it is sufficient to prove condition (i) for the edges of Q_i . To carry out a proof that Q_i P -covers Q_j we cover edges of Q_i by rectangles and verify that conditions (i) and (ii) hold. Generalized bisection technique is used to split the covering if it is necessary.

Fig. 9. (a) Covering of edges of quadrangles Q_i for the proof of existence of symbolic dynamics, (b) images of Q_1 , Q_2 , and Q_4 for the proof that Q_1 P -covers Q_7 , Q_2 P -covers Q_5 , and Q_3 P -covers Q_7 .

Using the method presented above, we have verified that each of the coverings present in (5) exist. For the computer assisted proof edges of Q_i have been covered by 618 rectangles, which are shown in Fig. 9(a). As an example, images of coverings of edges of Q_1 , Q_2 , and Q_4 are shown in Fig. 9(b). One can see that Q_1 P -covers Q_7 , Q_2 P -covers Q_5 , and Q_3 P -covers Q_7 . The proof of existence of all nine coverings took 16 seconds.

Summarizing, we have proved that the transition matrix (5) describes a valid symbolic dynamics between the sets Q_i . From Theorem 1 it follows that the topological entropy $H(P)$ of the return map P is not smaller than the dominant eigenvalue of A , i.e.:

$$H(P) \geq 0.1611. \quad (6)$$

Thus, we have shown that the topological entropy of the map P is positive, and the map P is chaotic in the topological sense.

Similar results can be obtained by analysing P_1 . Let us consider four quadrangles shown in Fig. 10(a). These are the same quadrangles as the four quadrangles located in the left half of Fig. 8.

There are five covering relations involving P_1 between these sets as described by the following transition matrix

Fig. 10. (a) Quadrangles Q_1 , Q_2 , Q_3 , and Q_4 for the proof of existence of symbolic dynamics for P_1 , (b) images of Q_1 , Q_2 ; Q_1 P_1 -covers Q_3 , and Q_2 P_1 -covers Q_4 .

$$A_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \quad (7)$$

Using the same technique as for the map P , we have rigorously proved that A_1 is a valid matrix describing covering relations involving P_1 between the sets Q_1 , Q_2 , Q_3 , and Q_4 . As an example in Fig. 10(b) we show coverings of Q_1 and Q_2 and enclosures of images of these coverings for the proof that Q_1 P_1 -covers Q_3 , and Q_2 P_1 -covers Q_4 . From Theorem 1 it follows that the topological entropy $H(P_1)$ of the return map P_1 is not smaller than the dominant eigenvalue of A_1 , i.e.:

$$H(P_1) \geq 0.3222. \quad (8)$$

This bound is equivalent to (8). This can be seen by noting that for $x \in Q_1 \cup Q_2 \cup Q_3 \cup Q_4$ we have $P(P(x)) = P_1(x)$ and that the topological entropy of the second iterate of the map is twice the topological entropy of the map itself, i.e. $H(P_1) = H(P \circ P) = 2 \cdot H(P)$.

6. Conclusions

The Muthuswamy-Chua memristor circuit has been analysed rigorously using interval analysis based computations. A trapping region containing the numerically observed attractor has been constructed. By proving the existence of a complex symbolic dynamics within this trapping region, we have shown that the return map associated with the memristor circuit is chaotic in the topological sense.

In future work, we plan to concentrate on finding symbolic dynamics with larger topological entropy, and study the existence and stability of limit sets (for example periodic orbits) belonging to the trapping region Ω .

Acknowledgments

This work was supported by the AGH University of Science and Technology, grant no. 11.11.120.343.

References

- [1] CAPD library, source code available at <http://capd.ii.uj.edu.pl/>.
- [2] G. Alefeld and J. Herzberger, *Introduction to interval computations*, Academic Press, New York, 1983.
- [3] L.O. Chua, "Memristor: The missing circuit element," *IEEE Trans. Circ. Theory*, vol. 18, no. 5, pp. 507–519, 1971.
- [4] L.O. Chua, "The fourth element," *Proc. IEEE*, vol. 100, no. 6, pp. 1920–1927, 2012.
- [5] L.O. Chua and S.M. Kang, "Memristive devices and systems," *Proc. IEEE*, vol. 64, pp. 209–223, 1976.

- [6] F. Corinto, A. Ascoli, and M. Gilli, “Nonlinear dynamics of memristor oscillators,” *IEEE Trans. Circuits Systems I*, vol. 58, no. 6, pp. 1323–1336, 2011.
- [7] F. Corinto, M. Gilli, A. Ascoli, and R. Tetzlaff, “Complex dynamics in neuromorphic memristor circuits,” In *Proc. European Conference on Circuit Theory and Design*, Dresden, 2013.
- [8] Z. Galias, “The dangers of rounding errors for simulations and analysis of nonlinear circuits and systems — and how to avoid them,” *IEEE Circuits and Systems Magazine*, vol. 13, no. 3, pp. 35–52, 2013.
- [9] Z. Galias and P. Zgliczyński, “Abundance of homoclinic and heteroclinic orbits and rigorous bounds for the topological entropy for the Hénon map,” *Nonlinearity*, vol. 14, pp. 909–932, 2001.
- [10] M. Itoh and L.O. Chua, “Memristor oscillators,” *Int. J. Bifurcation Chaos*, vol. 18, pp. 3183–3206, 2008.
- [11] C. Li, M. Wei, and J. Yu, “Chaos generator based on a PWL memristor,” In *Proc. Int. Conference on Communications, Circuits and Systems*, pp. 944–947, 2009.
- [12] Y. Li, L. Zhao, W. Chi, S. Lu, and X. Huang, “Implementation of a new memristor based chaotic system,” In *Proc. Int. Workshop on Chaos-Fractals Theories and Applications*, pp. 92–96, 2012.
- [13] R. Lohner, “Enclosing the solutions of ordinary initial and boundary value problems,” in *Computerarithmetic, Scientific Computation and Programming Languages*, pp. 225–286. Teubner, Stuttgart, 1987.
- [14] R.E. Moore, *Methods and applications of interval analysis*, SIAM, Philadelphia, 1979.
- [15] B. Muthuswamy, “Implementing memristor based chaotic circuits,” *Int. J. Bifurcation and Chaos*, vol. 20, pp. 1335–1350, 2010.
- [16] B. Muthuswamy and L.O. Chua, “Simplest chaotic circuit,” *Int. J. Bifurcation and Chaos*, vol. 20, no. 5, pp. 1567–1580, 2010.
- [17] A. Neumaier, *Interval methods for systems of equations*, Cambridge University Press, 1990.
- [18] R. Riaza, “First order mem-circuits: Modeling, nonlinear oscillations and bifurcations,” *IEEE Trans. Circuits Systems I*, vol. 60, no. 6, pp. 1570–1583, 2013.
- [19] C. Robinson, *Dynamical Systems: Stability, Symbolic Dynamics, and Chaos*, CRC Press, USA, 1995.
- [20] D.B. Strukov, G.S. Snider, D.R. Stewart, and R.S. Williams, “The missing memristor found,” *Nature*, vol. 453, pp. 80–83, 2008.
- [21] P. Zgliczyński, “Computer assisted proof of chaos in the Rössler equations and the Hénon map,” *Nonlinearity*, vol. 10, no. 1, pp. 243–252, 1997.
- [22] W. Zhang and Y. Yu, “Chaos generator based on a memristive circuit with time-delay feedback,” In *Proc. Int. Conference on Communications, Circuits and Systems*, pp. 817–819, 2010.