APPLICATIONS OF 1-D MAP FROM CHUA'S CIRCUIT: A PICTORIAL GUIDE

by

Marc Genot

Memorandum No. UCB/ERL M92/103

1 September 1992

APPLICATIONS OF 1-D MAP FROM CHUA'S CIRCUIT: A PICTORIAL GUIDE

by

Marc Genot

Memorandum No. UCB/ERL M92/103

1 September 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

APPLICATIONS OF 1-D MAP FROM CHUA'S CIRCUIT: A PICTORIAL GUIDE

by

Marc Genot

Memorandum No. UCB/ERL M92/103

1 September 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

This work is supported in part by the Office of Naval Research under grant N00014-89-J-1402 and the National Science Foundation under grant MIP-9001336.

Applications Of 1-D Map From Chua's Circuit : A Pictorial Guide

Marc Genot
Department of Electrical Engineering and Computer sciences
University of California, Berkeley, CA 94720, USA

08/25/92

Abstract

This paper is written as a *tutorial* on how to use a 1-dimensionnal map derived from Chua's circuit to study the circuit's complicated dynamics. While the derivation of this 1-D map is non-trivial, a user-friendly program is presented to help the beginner to uncover and witness, without any prior backround on chaos, numerous *periodic*, *homoclinic*, *heteroclinic* and *chaotic* orbits. In keeping with the pedagogical nature of this paper, these bifurcation phenomena will be profusely illustrated with pictures generated from a computer program, along with the exact parameters so that the reader can easily duplicate them. The program is written in the C-language for both PC-486 computers and UNIX workstations, and available upon requests from the Nonlinear Electronic Laboratory in Berkeley.

1 Introduction

Chua's circuit[1] is the *simplest* electronic circuit that exhibits the ubiquitous phenomenon called *chaos*. The history of the invention of this circuit is given in Reference 2. Because of its simplicity, robustness, and low cost[3], Chua's circuit, shown in Fig. 1, has become a pedagogical tool for studying and experimenting chaos. The state equations describing Chua's circuit are given in Reference 1. Since we will be concerned only with computer simulations in this paper, we will use this following simpler, equivalent, dimensionless equations[4] :

$$\begin{cases} \dot{x} = \alpha(y - x - f(x)) \\ \dot{y} = x - y + z \\ \dot{z} = -\beta y \end{cases} \tag{1}$$

where

$$f(x) = m_1 x + \frac{1}{2}(m_0 - m_1)[|x+1| - |x-1|] \tag{2}$$

is the equation describing the 3-segment piecewise-linear $v_R - i_R$ characteristic of the nonlinear resistor (Chua's diode[3]) in Fig. 1. For pedagogical purposes, it is convenient to assume :

$$\begin{aligned} &R = 1\Omega, \quad C_2 = 1F, \quad C_1 = \tfrac{1}{\alpha}F, \quad \text{and } L = \tfrac{1}{\beta}H \\ &x = v_{C_1}, \quad y = v_{C_2}, \quad \text{and } z = i_L \end{aligned} \tag{3}$$

Under these assumptions, the dimensionless Chua's circuit is described exactly by equations (1) and (2). Throughout this paper, we will assume :

$$m_0 = -8/7 \quad \text{and } m_1 = -5/7 \tag{4}$$

where m_0 is the slope of the outermost segment and m_1 is the slope of the middle segment of the v-i characteristic shown in Fig. 1(b). It is important to stress that because eq. (2) is a dimensionless equation, there is no loss of generality in our assumption (3) in the sense that for any behavior in the original circuit due to any choice of circuit parameters R, C_1, C_2, and L, there is a *corresponding* (α, β) where the dimensionless circuit behaves *identically.*

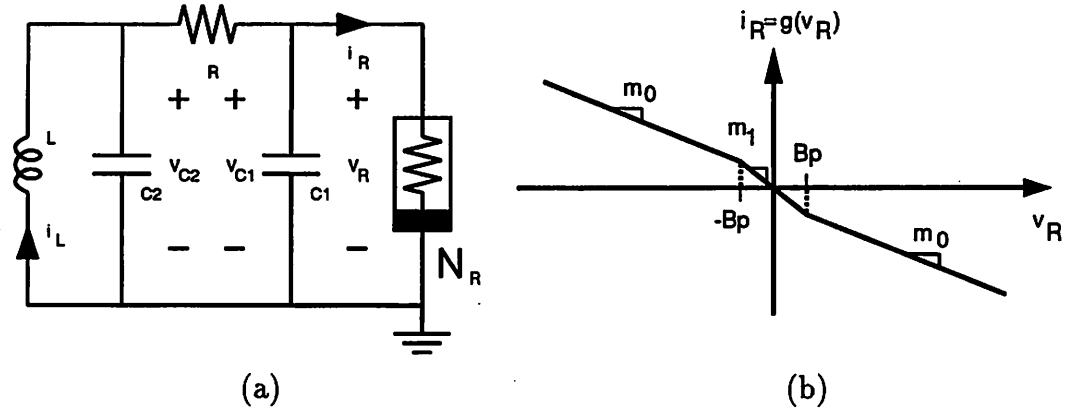


Figure 1: (a) Chua's circuit. (b) The v-i characteristic of the nonlinear resistor N_R.

Our goal in this paper is to use a 1-dimensional Poincaré map f, to be defined in Section 2, to help us pick strategic values of the parameters (α, β) which give rise to various qualitatively distinct behaviors in Chua's cicuit. The advantage of using a 1-D map is that the dynamical behavior of 1-D maps is much easier to explain and understand[5] compared to a brute force trial and error approach in the original system, while at the same time it offers a systematic method for choosing α and β.

For each choice of (α, β), we will also integrate the state equation (1) and show a projection of the corresponding waveform $(x(t), y(t), z(t))$, as well as the time waveform for $x(t)$. In most cases, we will see that the qualitative property of the solution $(x(t), y(t), z(t))$ agrees with what is predicted by an *inspection* of the 1-D map f with the same parameters α and β.

However, since the 1-D map f is derived from eq. (1) by making an *approximation*, there exist parameters (α, β) where the numerical solution of eq. (1) differs qualitatively

from what is predicted by f. We will provide some guidelines on when such discrepancies might occur. In such cases, the 1-D map f, though still interesting in its own right as a 1-dimensional dynamical system, can not be used as an exact model of Chua's circuit.

2 Description of the 1-D Poincaré map f and its Associated Software

A careful analysis[4] of the flow shows that in each of the outer regions of our piecewise-linear dynamical system, trajectories are strongly attracted towards the two dimensional unstable eigenspace of the unique outer equilibrium point P^+. In fact, the stable component of the vector field in this region is predominant and squeeze trajectories onto the unstable eigenspace. Though this result is not generic for Chua's circuit, it holds for the range of parameters to be investigated in this paper. Hence, the flow in the outer linear regions of the vector field is quickly confined in an infinitesimal neighborhood of the unstable eigenspace. Since a point in this volume is numerically indistinguishable from its projection onto the unstable eigenspace, it is therefore natural to construct an approximate 1-D Poincaré map f, of a line into itself, from an original 2-D Poincaré map defined on a selected[4] plane, transverse to this unstable eigenspace. Then, the domain of the map f lies on the intersection of the original Poincaré plane with the unstable eigenspace. Due to the geometric structure of the flow, it is possible to find an invariant *interval* for f. And, with a proper choice for the Poincaré plane, this 1-D map can be calculated efficiently, without solving any transcendental equations. We point out here that the equations of f obtained so far have a *parametric* form : we do not find an explicit formula for the function f.

To sum up, f is defined to be a single-valued function $y = f(x)$ from a closed interval, $[0, b]$ after rescaling, into itself:

$$f : [0, b] \longrightarrow [0, b] \tag{5}$$

where for the purpose of this section, f is generated by a computer program in the form of its graph, a non-monotonic curve C. For each value of (α, β), the program generates a corresponding curve $C(\alpha, \beta)$. The theory and exact algorithm implemented by the program are given in Section 5.

In this section, we will present some typical curves $C(\alpha, \beta)$ which gives rise to various distinct asymptotic behaviors. The uninitiated reader is referred to Reference 5 for introductory backround in 1-D maps.

2.1 Period-1 orbit

Figure 2(a) shows the curve C corresponding to $\alpha = 8$ and $\beta = 100/7$. Note that the fixed point X_1(intersection between C and the unit-slope line through the origin) is stable because the magnitude of the slope S of C at this point is less than unity.

The corresponding solution $\big(x(t), y(t), z(t)\big)$ of Eq. (1) is projected onto the X-Y plane

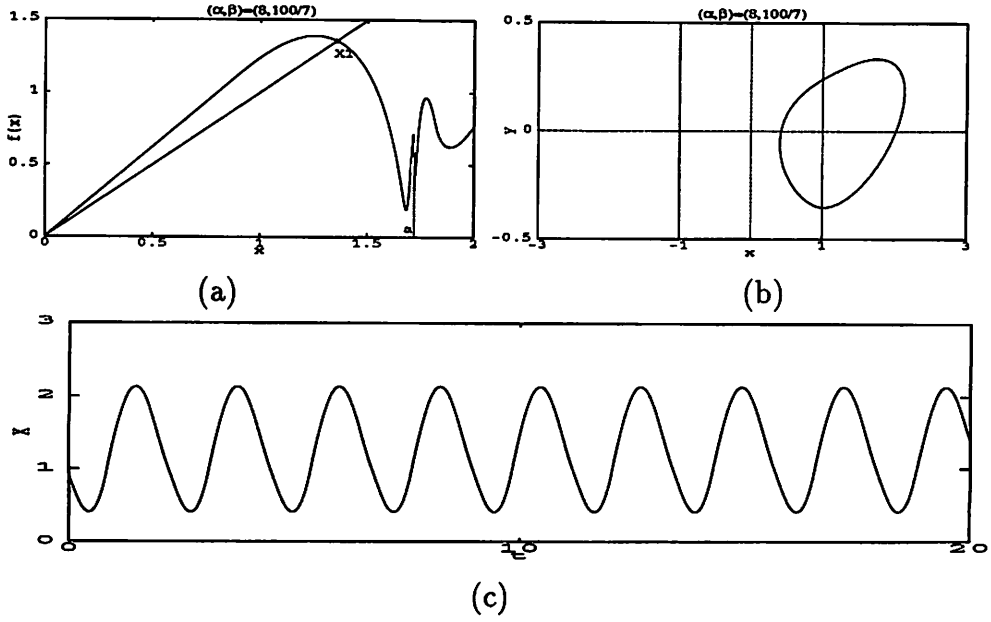


Figure 2: Stable period-1 orbit for $\alpha = 8$ and $\beta = 100/7$

in Fig. 2(b), and the time waveform for $x(t)$ is shown in Fig. 2(c). Using the theory from Appendix A, we can formulate the following operational interpretation, under the *standing assumption* that f is a realistic model of Chua's circuit.

Interpretation 1 *A stable fixed point of the 1-D map f implies the existence of a* stable *periodic solution in Chua's circuit, called a period-1 orbit.*

2.2 Period-3 orbit

Figure 3 shows the curve of the third iterate function f^3, corresponding to $\alpha = 8.58$ and $\beta = 100/7$. This picture is generated from an algorithm included in the computer program presented with this paper (see Section 5). A fixed point, $\overline{x}$, for the map f^3 which is not a fixed point for f is called a period-3 *point* for the Poincare map f. The orbit of f based on such a point $\overline{x}$ visits the three-point sequence $\{\overline{x}, f(\overline{x}), f^2(\overline{x})\}$ before mapping onto $f^3(\overline{x}) = \overline{x}$. This sequence is called a period-3 *orbit* for our 1-D map and each of its component is necessarily a fixed point for f^3.

By the chain rule of differentiation, we obtain :

$$f^{3'}(\overline{x}) = f'\big(f^2(\overline{x})\big) f'\big(f(\overline{x})\big) f'\big(\overline{x}\big)$$

It follows that the derivative of f^3 is the same at $\overline{x}$, $f(\overline{x})$ or $f^2(\overline{x})$. One could use this simple property to find out the period-3 orbits of the 1-D map f from the plot of f^3 and by the way determine the stability of this periodic orbit. Among the period-3 sequences

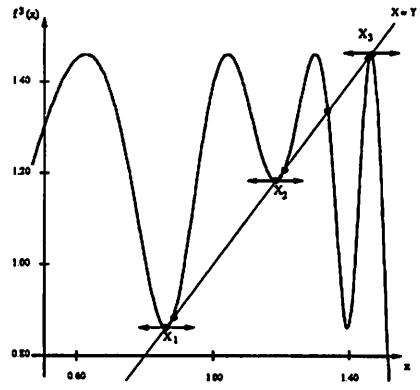


Figure 3: For $\alpha = 8.58$ and $\beta = 100/7$, the 1-D map f is iterated three times to obtain the plot of f^3. The almost horizontal segments represent the slope of f^3 at the 3 stable fixed points. The other fixed points of f^3 have slopes greater than unity and are therefore unstable.

exhibited in Fig. 3, note that the sequence $\{X_1, X_2, X_3\}$ is stable because the magnitude of the slope of the curve of f^3 at these points is less than unity.

Figure 4(a) shows the corresponding period-3 orbit on the curve $C(\alpha, \beta)$, with a schematization of the sequence of points visited by the orbit. The corresponding solution $\big(x(t), y(t), z(t)\big)$ of Eq. (1) is projected onto the X-Y plane in Fig. 4(b), and the time waveform for $x(t)$ is shown in Fig. 4(c). Still using the theory from Appendix A, we can formulate the following operational interpretation, under the *standing assumption* that f is a realistic model of Chua's circuit.

Interpretation 2 *A stable period-3 sequence of the 1-D map f implies the existence of a* stable *periodic solution in Chua's circuit.*

With the same standing assumption, provided by the theoretical approach presented in Appendix A, the same interpretation can be extended to any periodic sequence of the 1-D map f.

Here we mention the fascinating result obtained by Li and Yorke[6] in 1975. If a 1-D continuous map of an interval into itself has a stable or unstable period-3 orbit, then there is an uncountable set of initial points for which the flow leads to a chaotic motion (see Section 2.5 for a presentation of another kind of chaos).

However, this *particular* chaotic behavior is generally physically unobservable[7] because for most -in the sense of Lebesgue measure- other points, the system is not necessarily chaotic. In fact, both a computer simulation of eq. (1) with $\alpha = 8.58$ and $\beta = 100/7$, and an experimental realization of Chua's circuit show that most initial points on the interval $[0, b]$ converge toward the unique stable period-3 orbit of the dynamical system.

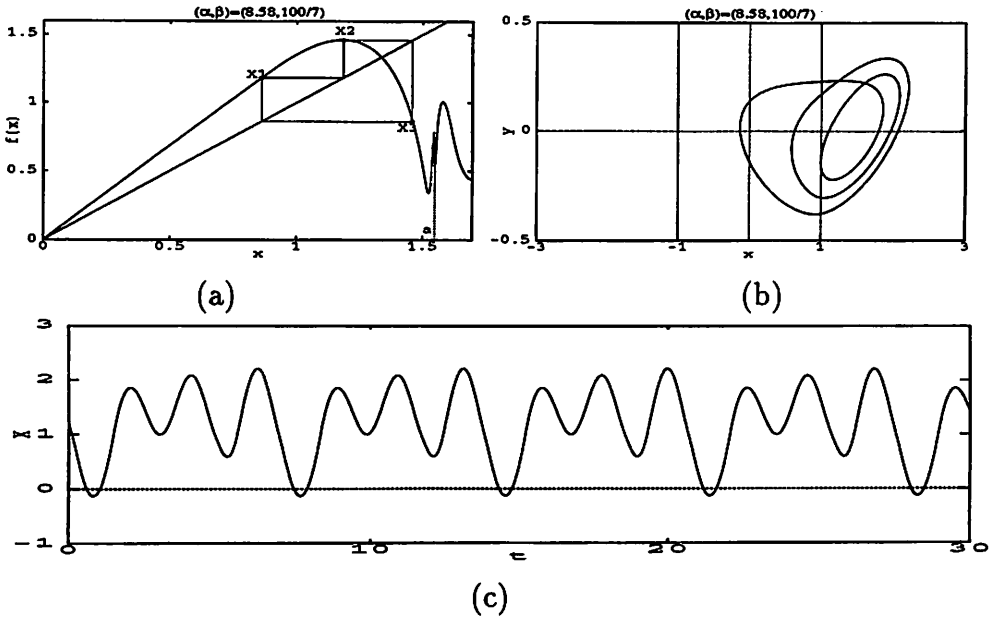


Figure 4: Stable period-3 orbit for $\alpha = 8.58$ and $\beta = 100/7$

2.3 Homoclinic orbits

A homoclinic orbit for the dynamical system (1) is a trajectory connecting an equilibrium point to itself. This equilibrium point is of saddle-focus type in Chua's circuit. Fig. 5(b) presents an image of a homoclinic orbit in Chua's circuit. The fixed point O has a one-dimensional unstable manifold W^U (along the vector associated with a real eigenvalue of the linear flow) and a two-dimensional stable manifold W^S (the plane associated with a complex conjugate pair of eigenvalues of the flow, with negative real parts). As shown

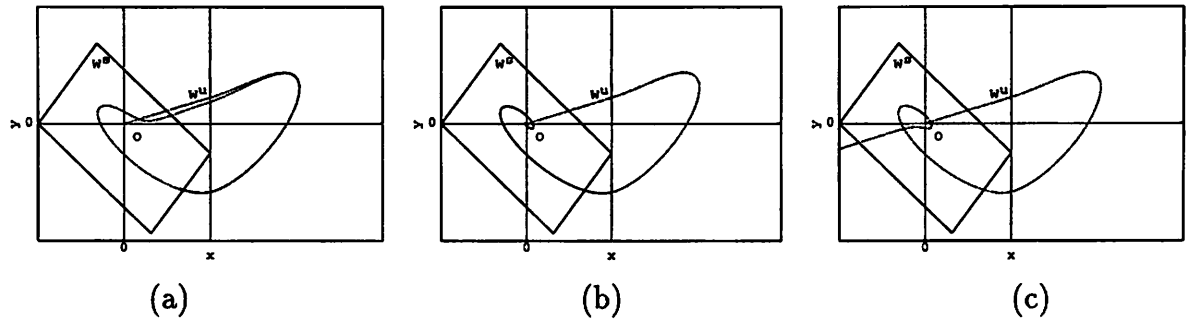


Figure 5: Structural instability of homoclinic orbits in R^3. (b) Homoclinic orbit for the parameter set $(\alpha_0, \beta_0) \simeq (11.0917459, 100/7)$. (a) and (c) Behavior of the unstable manifold of the fixed point O for $\alpha < \alpha_0$ and $\alpha > \alpha_0$ respectively.

in this figure, the trajectory based at the origin first advances on the unstable manifold W^U while in the middle region. Then its continuation in the outer region re-enters the central region on the stable manifold W^S and keeps looping on it as time tends to infinity.

The trajectory cannot intersect the plane W^S transversaly, as a direct consequence of uniqueness of solutions in *autonomous* systems.

As a general rule, a homoclinic orbit forms a closed loop like a periodic orbit, but unlike a periodic orbit, it takes infinitely long time to traverse the loop. The orbit approachs the stationary point in both forwards or backwards time but never reaches it in a finite time, in each time direction. However, as stressed in Section 4, it can still be thought of as a limiting case of periodic orbits for our particular system.

A homoclinic orbit is destroyed by a general perturbation of the dynamical system in which it arises. This property, known as the *structural instability* of homoclinic orbits in R^3, comes from the non-transversality condition mentioned above. As depicted in Fig. 5, a slight variation on the set of parameters which gives rise to a homoclinic orbit dramatically changes the qualitative behavior of the flow, and the homoclinicity does not hold any more. This remark, combined with the fact that the fixed point visited by the homoclinicity is of saddle type, makes it virtually impossible to observe any such orbit both experimentally, or through extensive computer simulations.

However, it remains possible for Chua's circuit to establish the *existence* of homoclinic orbits. Having in mind the global 2-parameter bifurcation structure of eq. (1) briefly presented in Section 4, one can even easily localize several homoclinic orbits with the help of our 1-D map f.

2.4 Heteroclinic orbits

A heteroclinic orbit is the union of distinct fixed points of the flow, and the trajectories connecting them. Unlike in the case of a homoclinic orbit, this union does not necessarily form a closed loop in the phase space. The trajectories belonging to the heteroclinic

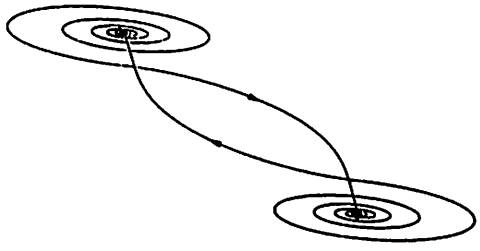


Figure 6: Heteroclinic orbit.

orbit approach the two stationary points they connect in either forwards or backwards time as time tends to infinity. As shown in Fig. 6, any such trajectory is both included in the unstable manifold of one fixed point and the stable manifold of the other fixed point, in each linear region separately.

A heteroclinic orbit in eq. (1) disappears throught an arbitrarily small perturbation and is thus structurally unstable in Chua's circuit.

Like homoclinic ones, the heteroclinic orbits of our dynamical system are experimentally and physically unobservable, as a direct consequence of the saddle-focus nature

of the fixed points and the structural instability of its heteroclinic orbits. But unlike homoclinic orbits for which the knowledge of the global 2-parameter bifurcation structure allowed their localization with the only help of the 1-D map f (see Section 4), heteroclinic orbits are very difficult to localize.

2.5 Chaotic orbit

A pragmatic characterization of a chaotic trajectory is an *unpredictable* and *bounded* motion. The limit set for chaotic trajectories is not a simple geometric object such as an equilibrium point, a periodic or a quasiperiodic trajectory but is related to fractal and Cantor sets. A chaotic trajectory accumulates on a so-called *strange attractor*. The name 'strange attractor' refers to its surprising properties. The crucial one is the so-called sensitive dependence on initial conditions : two trajectories of the attractor initially infinitesimally close to one another exponentially diverge as time increases (in practice, an initial condition is always specified within a tolerance interval). In other words, the auto-correlation function becomes equal to zero after a finite amount of time. Therefore, no matter how precisely the initial condition is known, the asymptotic behavior of a chaotic trajectory is *unpredictable*.

Two examples of chaotic trajectories exhibited in Chua's circuit are shown in Fig. 7 : the Double Sroll attractor and the Rössler-type attractor. It is obvious from these pic-

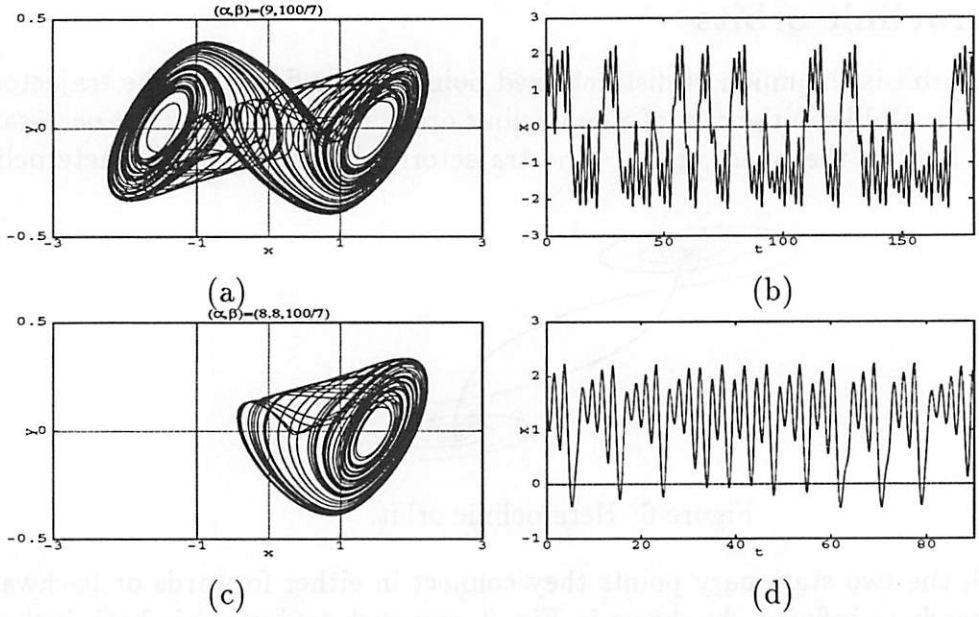


Figure 7: Chaotic trajectories. (a) Double Scroll attractor. (b) Time waveform of the x component of (a). (c) Rössler-type attractor. (d) Time waveform of the x component of (c).

tures that for these two sets of parameter's values, the trajectories are, indeed, bounded and aperiodic. Rigorously, we would need some additional information (the spectrum

of the x component of the trajectory for example) to determine whether the solution is quasiperiodic or not.

The reader is referred to Reference 4 for a rigorous mathematical proof of the *chaotic nature* of the Double Scroll attractor, derived from a direct application of Shilnikov's theorem. Also, some experimental results such as the calculation of Lyapunov exponents or fractal dimensions confirm the occurence of robust chaos in Chua's circuit.

3 Catalog of all period-n orbits, $n = 1, \ldots, 5$

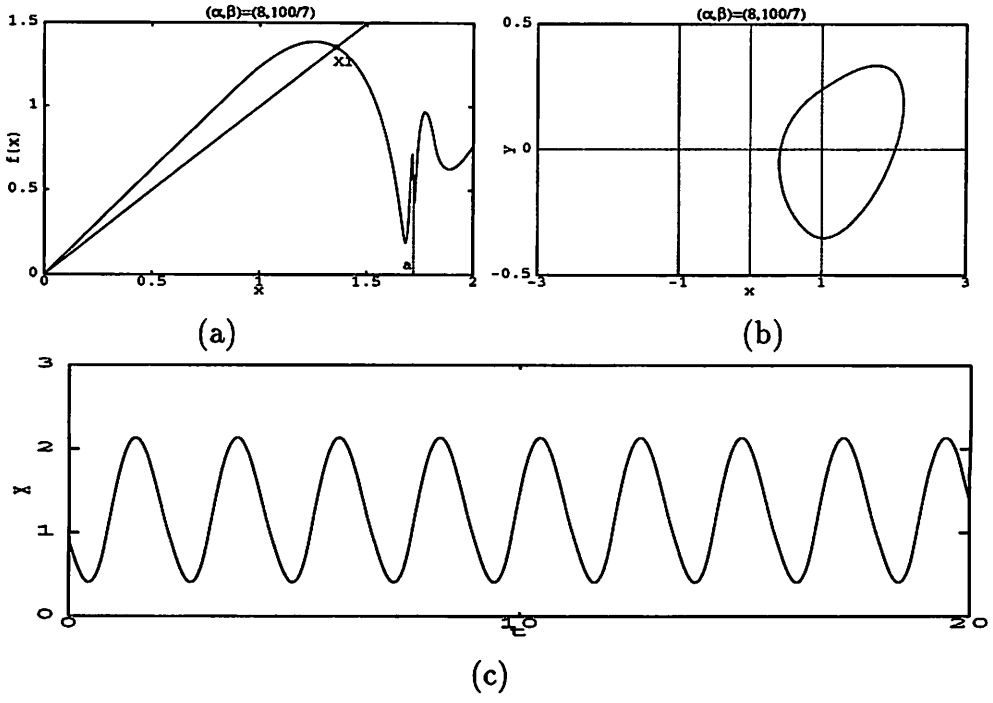
Here we present an exhaustive catalog of a wide variety of stable periodic orbits exhibited by the 1-D map f of Chua's circuit, for different parameter values. Here, the classification of a periodic orbit is determined by the order of the periodic sequence of f, and the position of the critical point a (a on the interval $[0, b]$) relatively to the components of this periodic sequence. Briefly speaking, the critical point a is a *gate* beyond which the trajectory emanating from an initial point on the interval $[0, b]$ crosses the two separation planes of the dynamical system (1) (namely the planes $\{(x, y, z) : x = \pm 1\}$) before reaching the first return point on the domain of our 1-D map. Instead, for an initial point below a, the trajectory crosses only one discontinuity. For each type of periodic sequence, a three-part figure is provided, with a set of exact values for α and β. Sometimes the standing assumption leading to interpretation 1 and 2 in section 2 does not hold : to a stable periodic orbit of f may correspond an unstable periodic orbit in the original 3-D system (see Appendix A). In such a case, the unstable periodic trajectory is ploted with a dotted curve instead of a solid curve for stable periodic trajectoties. The parameter values have been arbitrarily chosen among non-empty *intervals* of the control parameter's space (α, β) for which the 1-D map f is known to contain the same type of attractor. In this section, we use the following convention for each figure : (a) shows the curve $C(\alpha, \beta)$, (b) the projection of the solution $\Big(x(t), y(t), z(t)\Big)$ of eq. (1) onto the X-Y plane and (c) the time waveform for $x(t)$.

As the reader shall notice, the period orders of the orbits of the 1-D map may differ from those of the real 3-D system described by eq. (1). These differences come from the geometrical nature of the flow and the construction of the 1-D map. In a periodic sequence of f, we count the number of revolutions performed by the original trajectory in the *upper* region ($\{(x, y, z) : x \geq 1\}$) of the piecewise-linear vector space, while ignoring, apparently, the motion of this trajectory in the two other regions. For example, for $\alpha = 10.49$ and $\beta = 100/7$, the 1-D map f has a stable *period-3* orbit which correspond to a *period-2* trajectory in the original system.

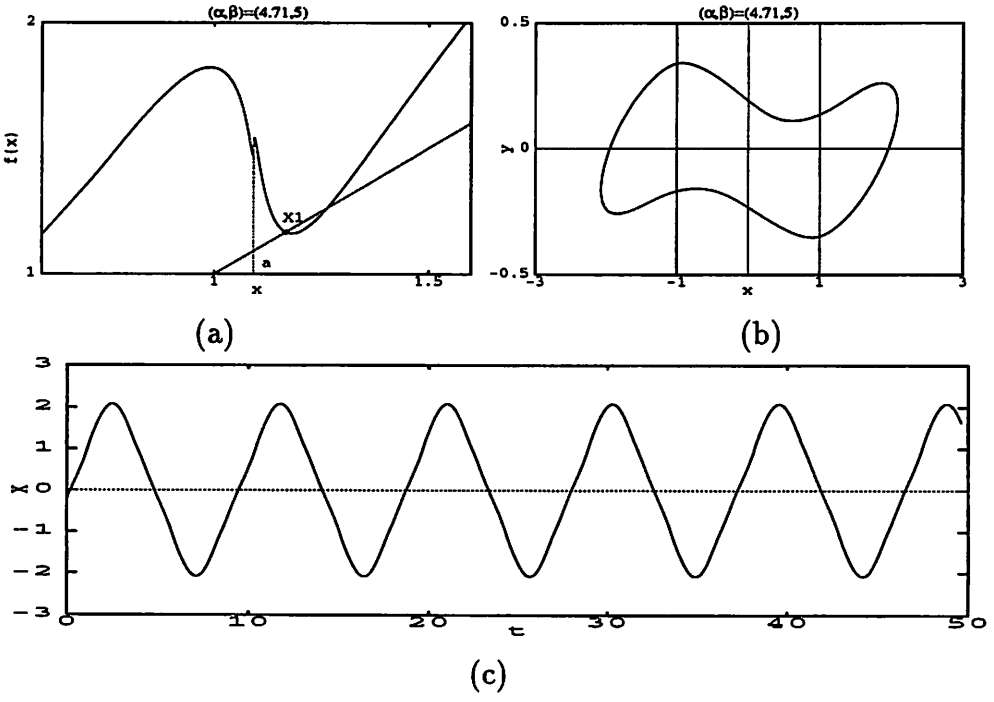
Furthermore, due to the symmetry of the vector field with respect to the origin, each time the displayed periodic trajectory (b) is asymmetric, a second periodic trajectory exists, which is the odd-symmetric image of the ploted trajectory. Here, we do not represent them in order to make the plots more clear.

3.1 Period-1 orbits

A)

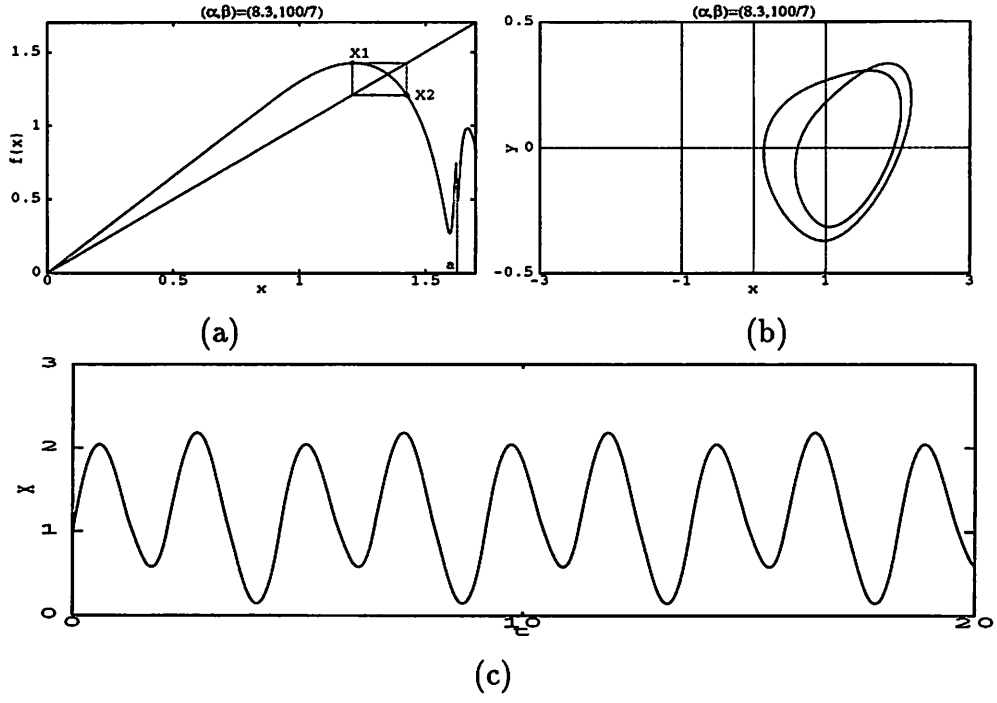


B)

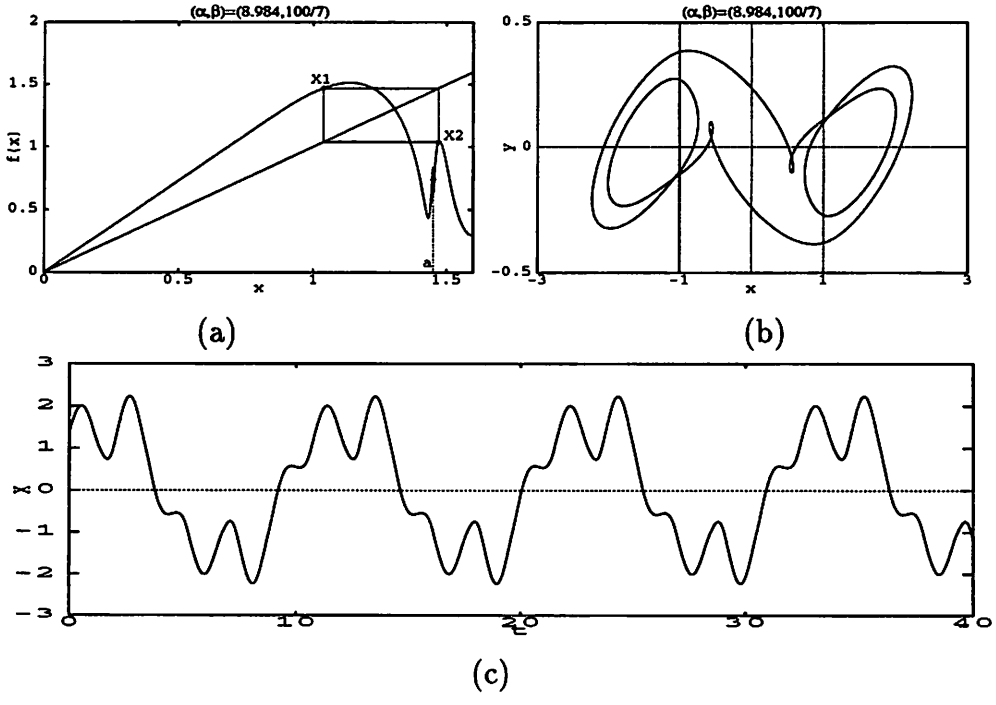


3.2 Period-2 orbits

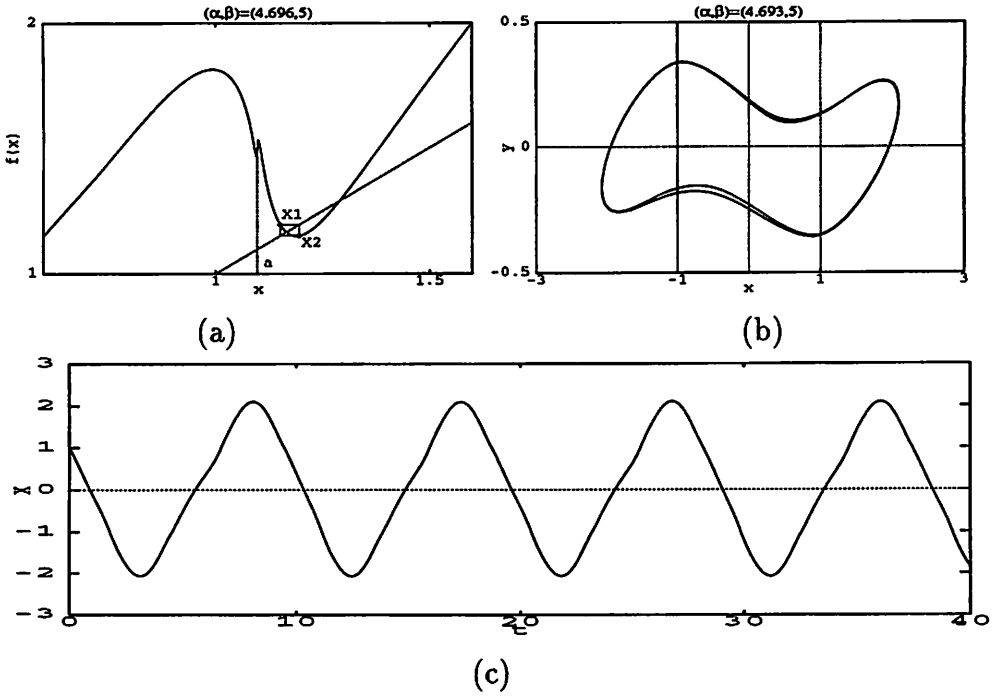
A)



B)

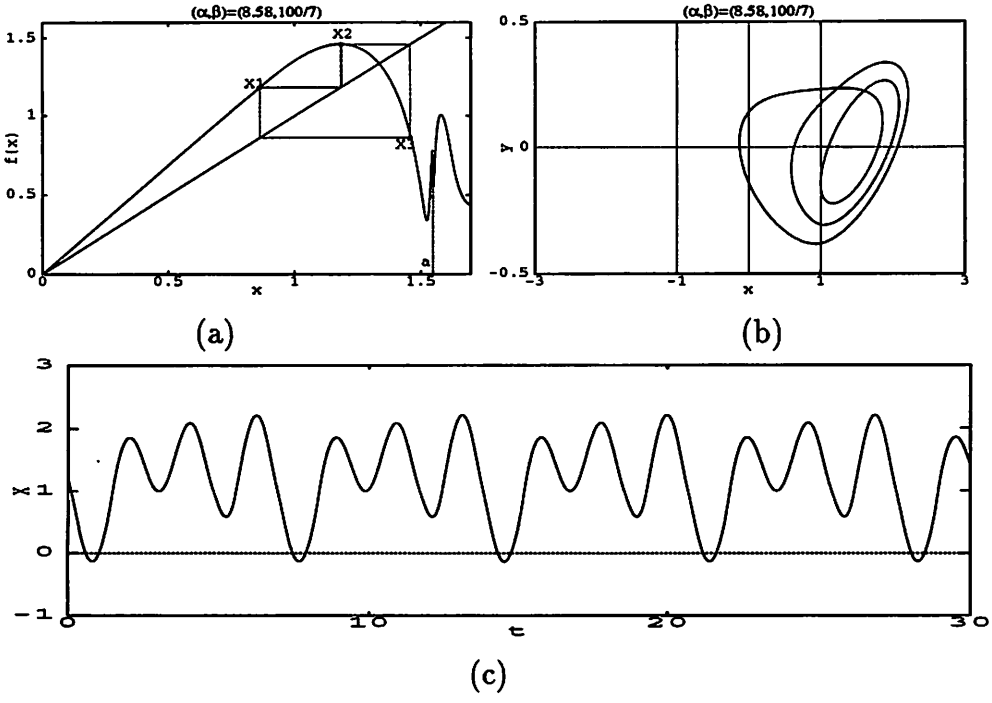


C)

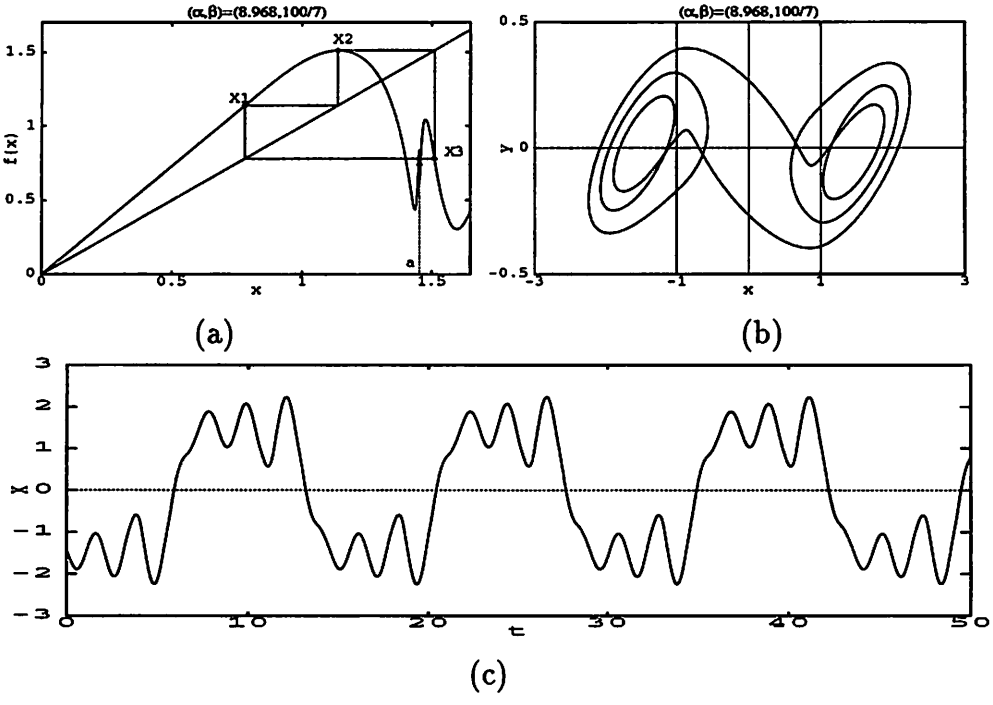


3.3 Period-3 orbits

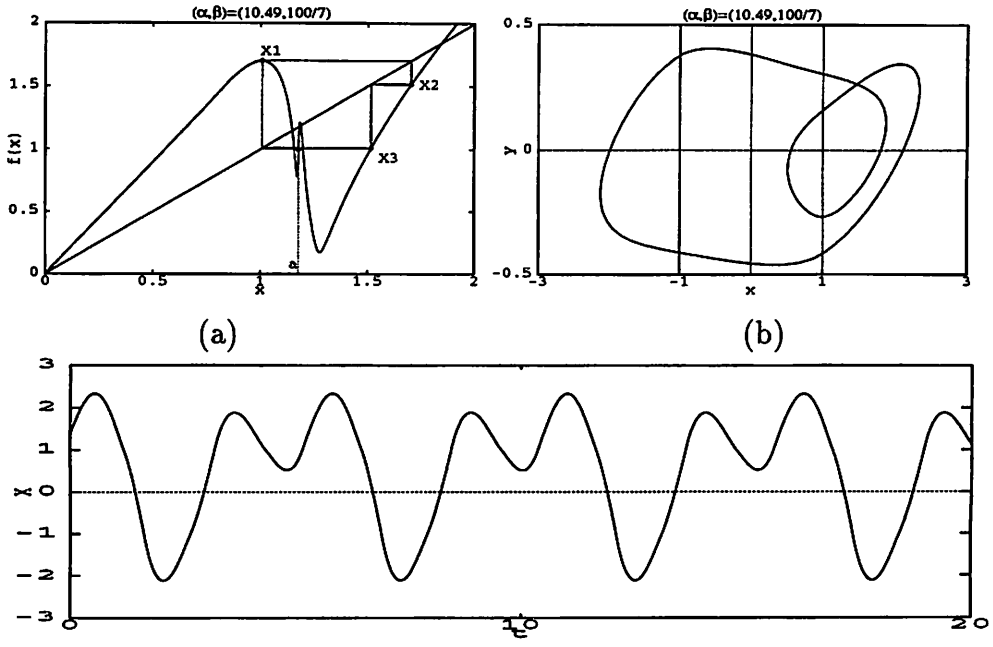
A)



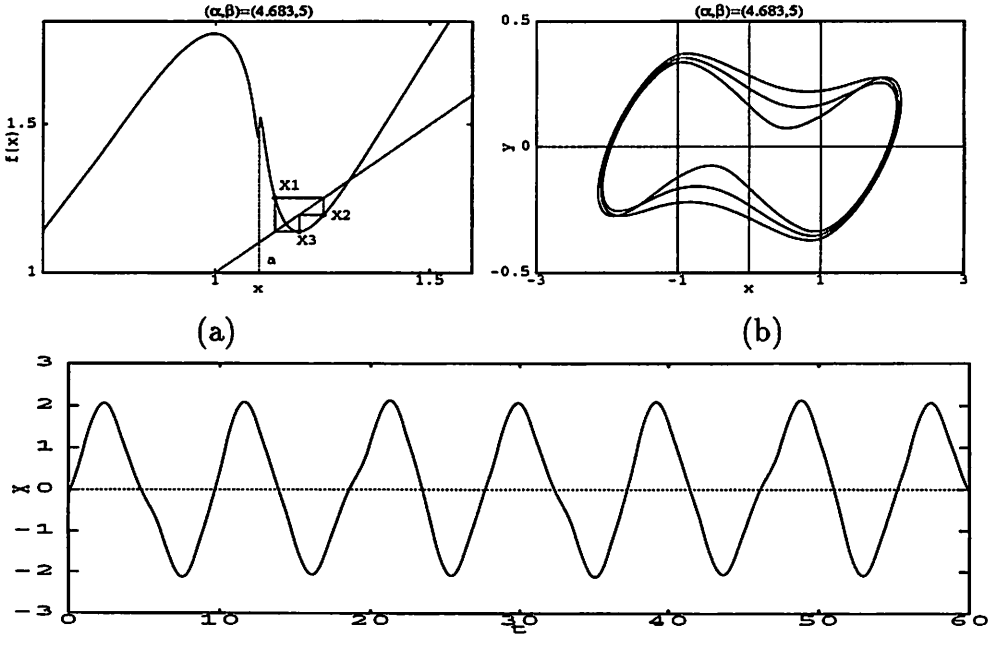
B)



C)

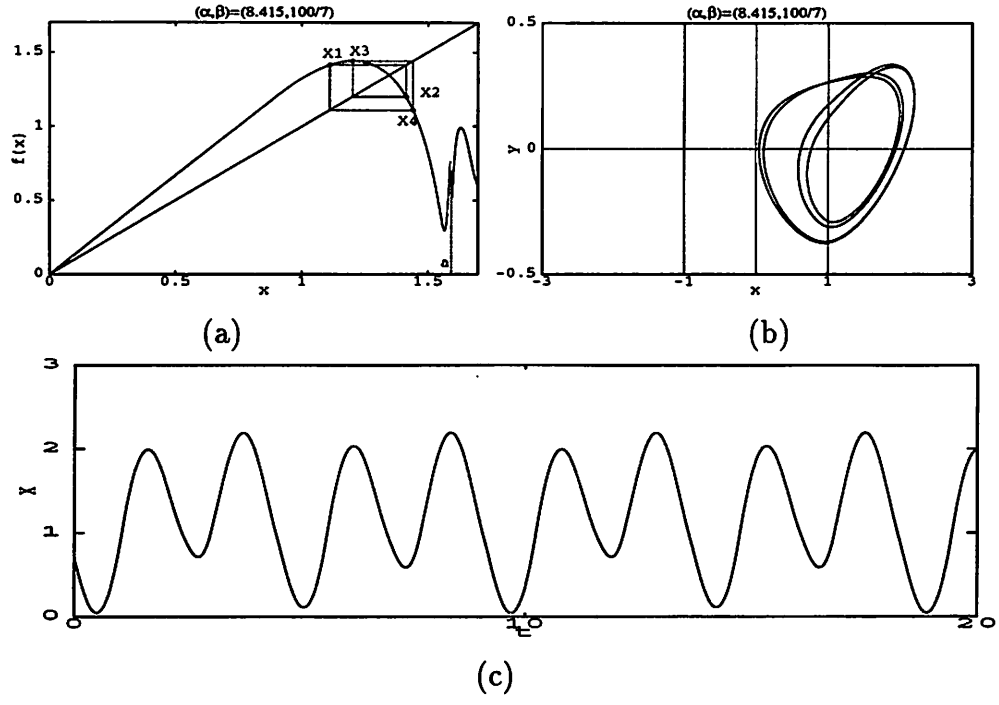

(α,β)=(10.49,100/7)
X1
X2
X3
f(x)
x
a
(α,β)=(10.49,100/7)
y
x
(a)
(b)
X
t
(c)

D)

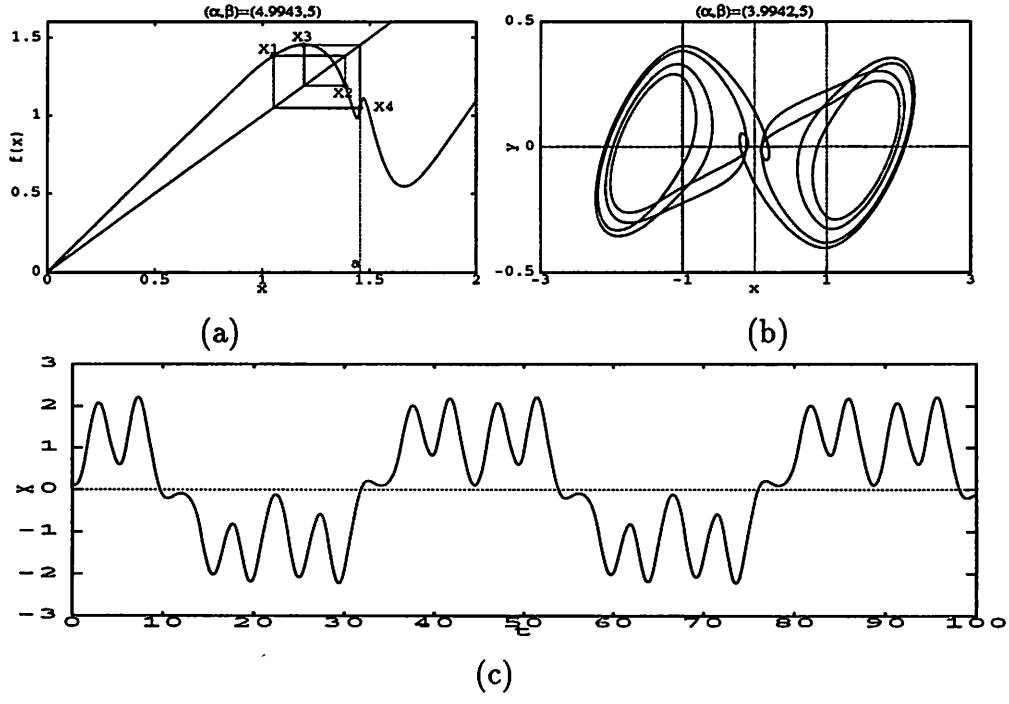

(α,β)=(4.683,5)
X1
X2
X3
f(x)
x
a
(α,β)=(4.683,5)
y
x
(a)
(b)
X
t
(c)

3.4 Period-4 orbits

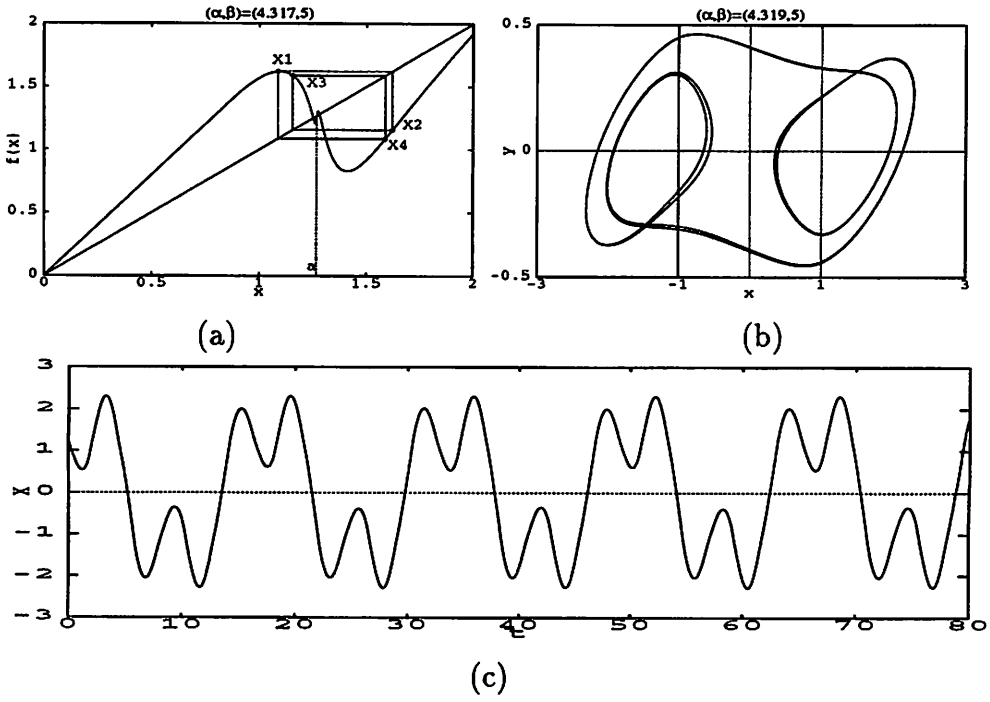
A)



B)



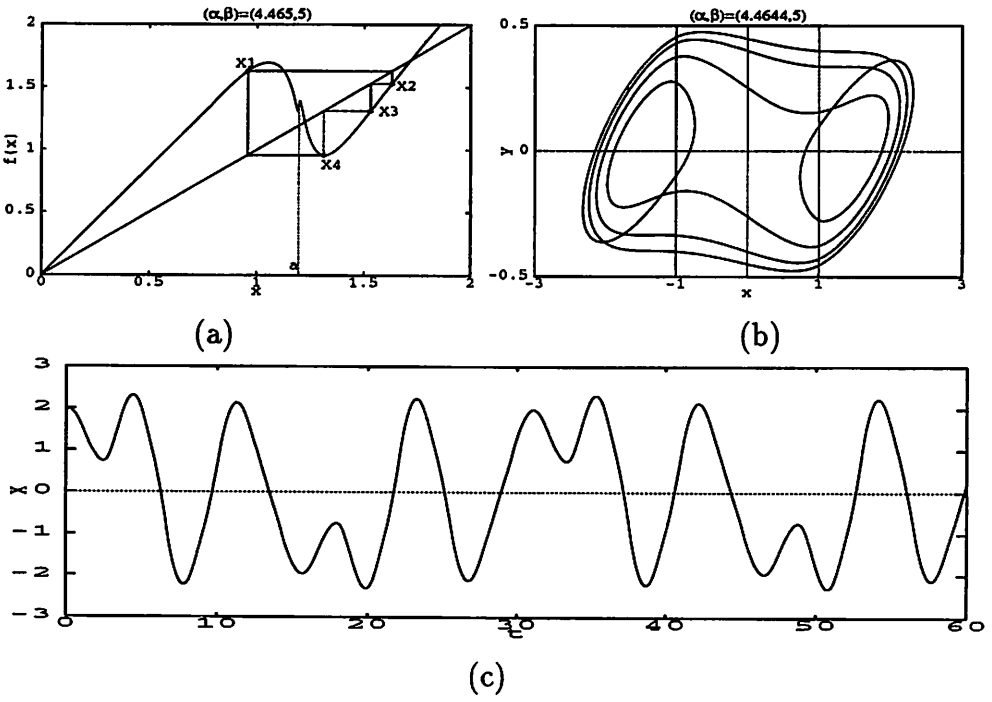
C)


(α,β)=(4.317,5)
X1
X3
X2
X4
f(x)
x
(α,β)=(4.319,5)
y
x
X
t

(a) (b)

(c)

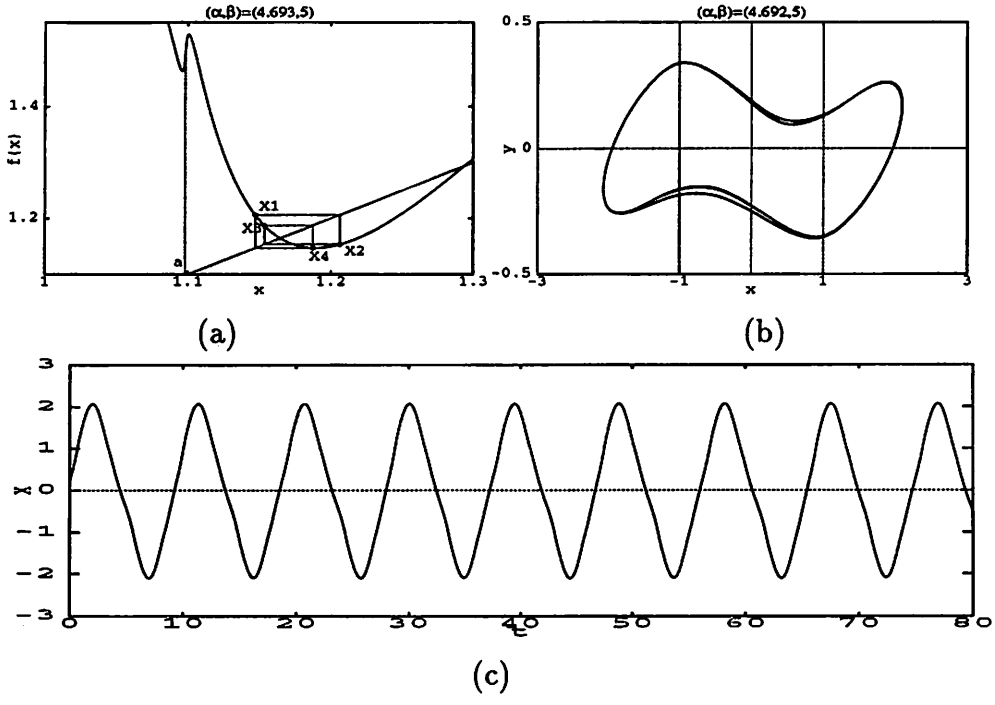
D)


(α,β)=(4.465,5)
X1
X2
X3
X4
f(x)
x
(α,β)=(4.4644,5)
y
x
X
t

(a) (b)

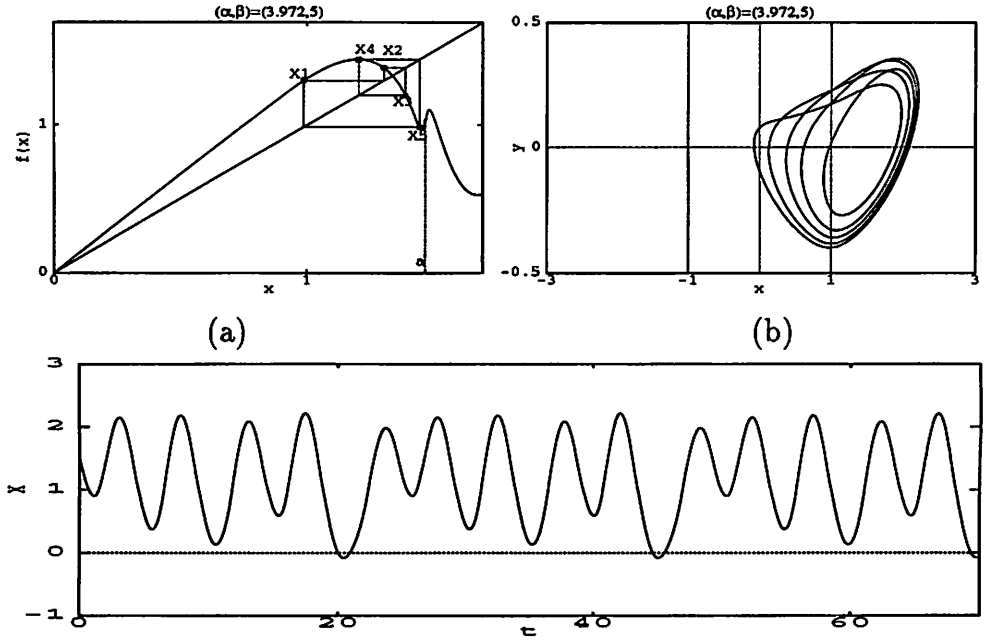
(c)

E)

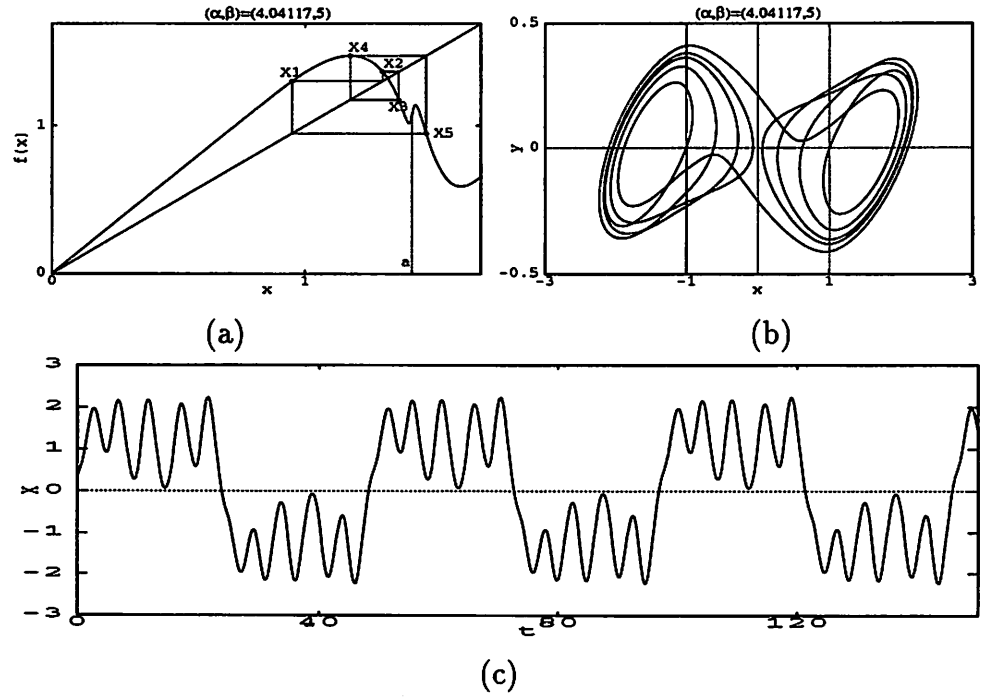


3.5 Period-5 orbits

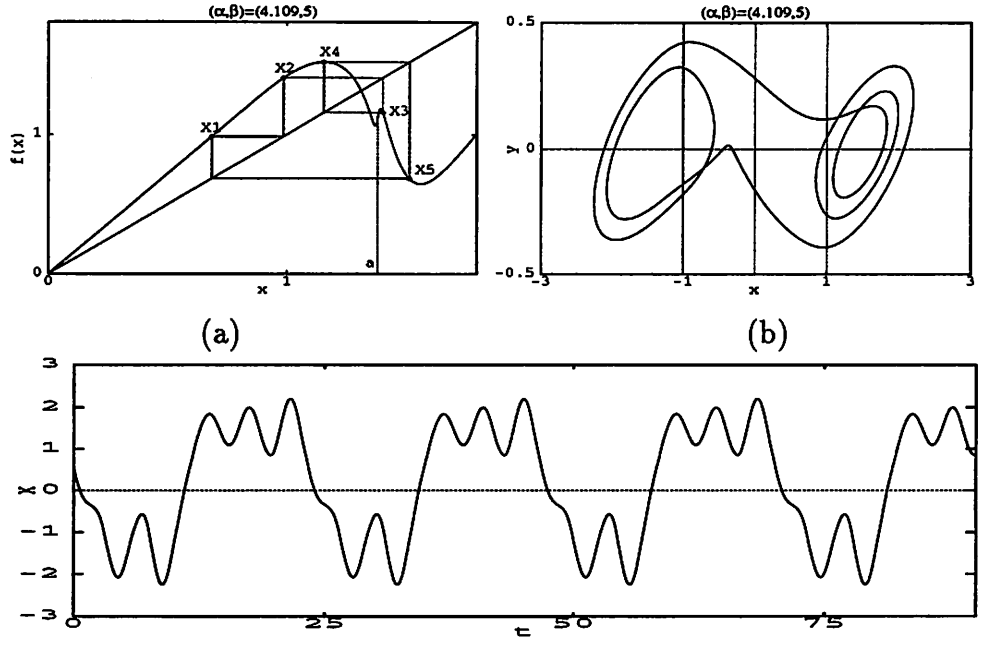
A)



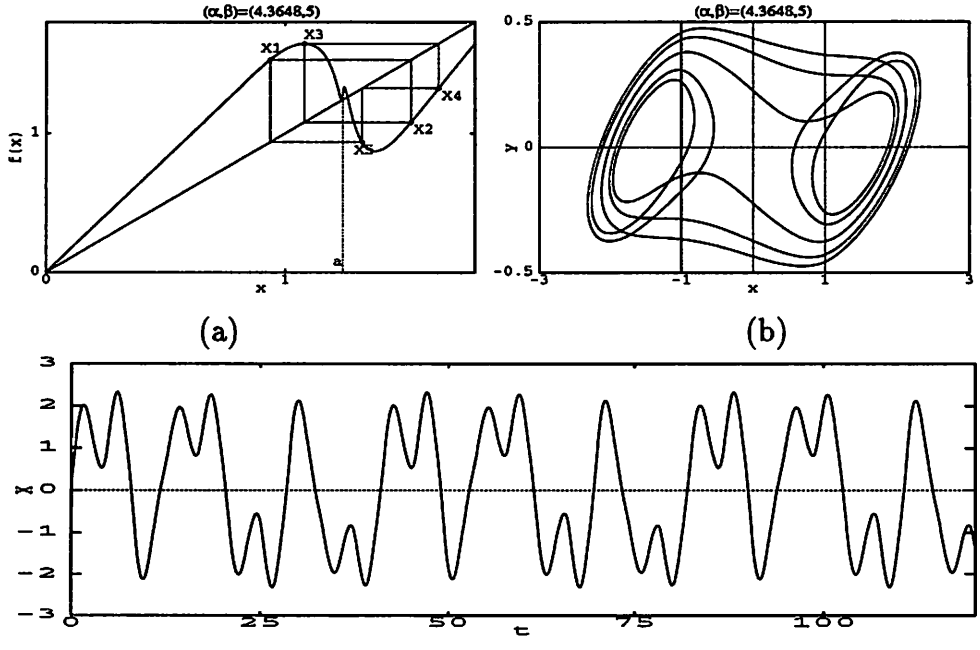
B)



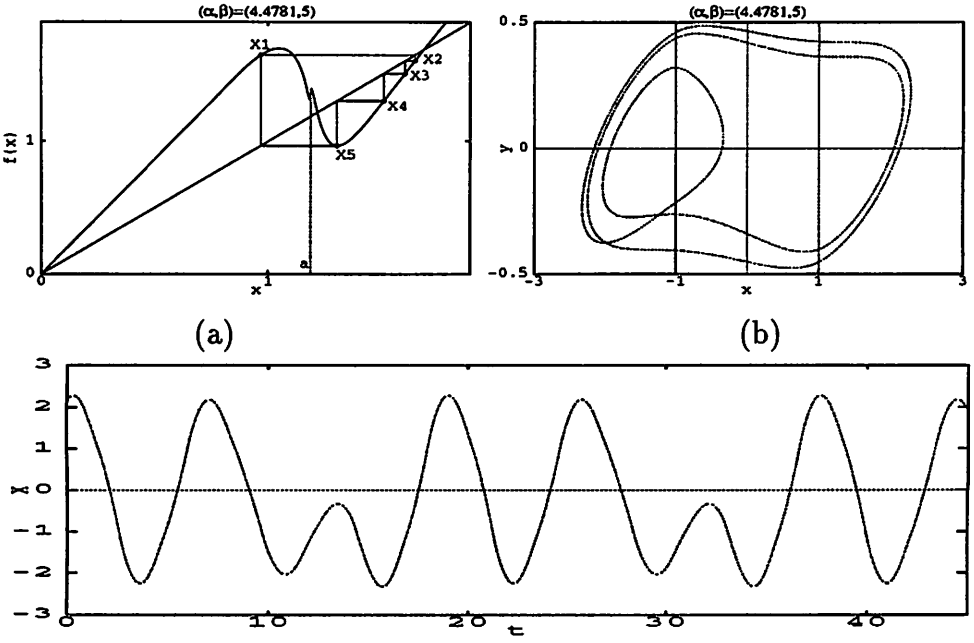
C)


(α,β)=(4.109,5)
X1
X2
X3
X4
X5
f(x)
x
a
(α,β)=(4.109,5)
y
x
(a)
(b)
X
t
(c)

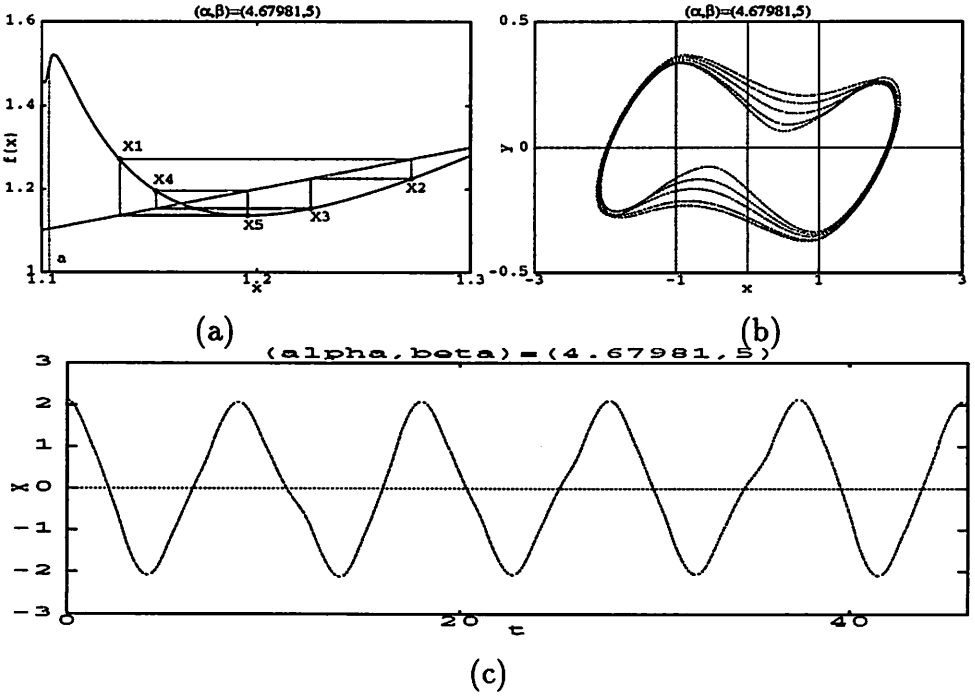
D)


(α,β)=(4.3648,5)
X1
X3
X4
X2
X5
f(x)
x
a
(α,β)=(4.3648,5)
y
x
(a)
(b)
X
t
(c)

E)


(α,β)=(4.4781,5)
X1
X2
X3
X4
X5
f(x)
x
a
(a)
(α,β)=(4.4781,5)
y
x
(b)
X
t
(c)

F)


(α,β)=(4.67981,5)
X1
X4
X2
X5
X3
f(x)
x
a
(a)
(α,β)=(4.67981,5)
y
x
(b)
(alpha,beta)=(4.67981,5)
X
t
(c)

4 One-parameter bifurcation of asymmetric orbit Π_0^1 and symmetric orbit Π^1

In this section an analysis of the bifurcation phenomena occuring in Chua's circuit is presented in order to make the occurence of periodic limit cycles in this dynamical system easier to understand and provide a general comprehension of the mechanisms of orbit creation.

In the previous sections, our concern was to establish the consistency of our approximation in predicting the existence of periodic trajectories in the 3-D system (1). Here, we focus our attention on explaining the complex structure of periodicities in eq. (1), for numerous windows in the parameter space. At this point this complexity is mostly suggested by the high variety of periodic orbits of all order observed on the graph of f for parameter values in these windows. The consistency of our approximation guarantees the validity of this interpretation. Hence, for the purpose of this section, a one-parameter bifurcation diagram of Chua's circuit is presented in Fig. 8 (we fix $\beta = 100/7$ while varying α). The main contribution of this diagram is to provide valuable information on the approximate 1-D map f as well as on its bifurcation structure.

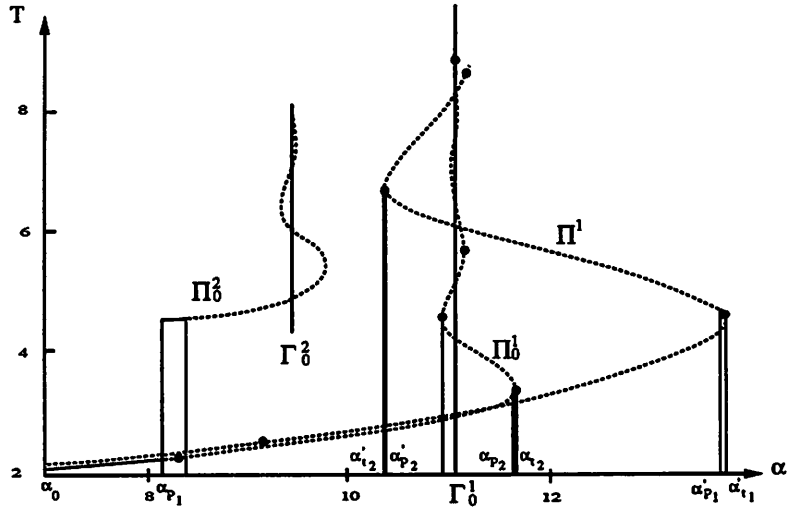


Figure 8: Experimental bifurcation curves of the periodic orbits $\prod_0^1$ and $\prod^1$. The deformation of these orbits as they approach the homoclinicity Γ_0^1, is illustrated in Fig. 9. The unstable sections of the curves are plotted with dotted lines and the stable ones with plain lines. Here the periodic orbits are represented with their full period, and not the usual half period in the literature.

The dynamics of system (1) has a fixed point in each linear region : the origin O of saddle-type focus, and P^+ and P^- located in the two outer regions, and symmetric with respect to the origin. In the central region O has a two-dimensional unstable manifold and a one-dimensional stable manifold. For $\alpha < \alpha_0$, P^+ and P^- are sinks.

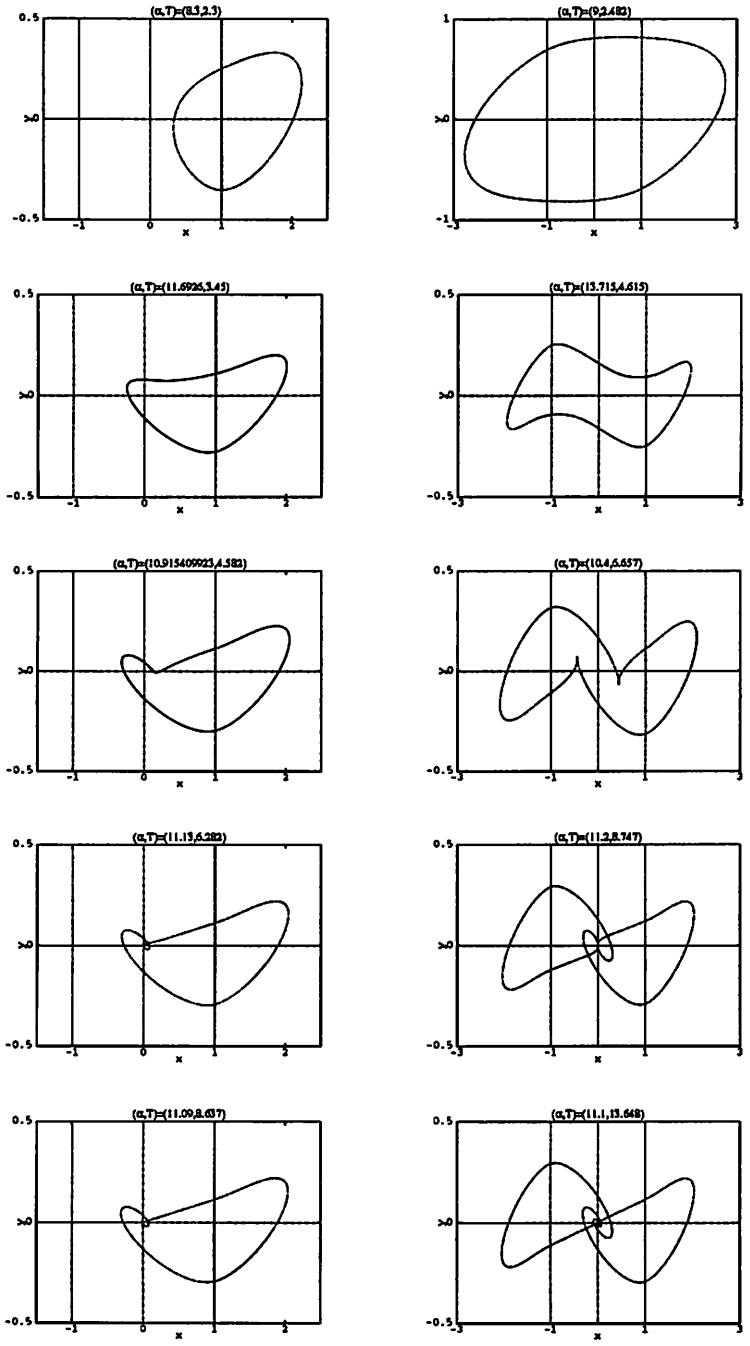


Figure 9: The two orbits $\prod_0^1$ and $\prod^1$ illustrated at the consecutive solid dot points on Fig. 8 : (a) for the periodic orbit $\prod_0^1$ and (b) for $\prod^1$. The stable periodic orbits are plotted with solid curves, and the unstable ones with dotted curves.

A Hopf-like bifurcation occurs at $\alpha = \alpha_0 \simeq 7.0$, giving rise to a pair of nonsymmetric stable periodic orbits $\prod_0^1$ and a symmetric unstable periodic orbit $\prod^1$. Fig. 8 shows how these periodic trajectories evolve in the parameter space α (the ordinate gives the period of the trajectory). The orbit $\prod_0^1$ loses stability in a period-doubling bifurcation at $\alpha = \alpha_{p_1}$, giving rise to a period-2 orbit $\prod_0^2$ which then bifurcates at $\alpha = 8.1639$ in a period doubling cascade process. After that, the orbits $\prod_0^1$ and $\prod^1$ wind around $\alpha = \Gamma_0^1 \simeq 11.0917$, while period tends to infinity and amplitude of oscillations decreases as they approach Γ_0^1. The periodic orbits converge towards the homoclinic orbit Γ_0^1 in a so-called homoclinic bifurcation. This behavior has been first observed in Chua's circuit by George[10] while the theory of this bifurcation is due to Glendinning and Sparrow[11]. Fig. 9 shows the continuous deformation of $\prod_0^1$ and $\prod^1$ into the homoclinic orbit Γ_0^1.

In this process, $\prod_0^1$ and $\prod^1$ successively lose and regain their stability through saddle-node $(\alpha_{t_1}', \alpha_{t_2}', \alpha_{t_2})$ and pitchfork bifurcations $(\alpha_{p_1}', \alpha_{p_2}', \alpha_{p_2})$. The stable sections of the curves appear to be very short and even numerically indistinguishable as we approach the homoclinicities. The latter bifurcations and the period-doubling bifurcations also produce other periodic orbits of higher orders for which we expect a behavior similar to $\prod_0^1$ and $\prod^1$ in the parameter space. We refer the reader to Reference 12 for further details on the bifurcation diagram of Chua's circuit.

Through all simulations performed so far, the periodic orbits of the 1-D map f itself behave identically to the periodic trajectories of eg. (1) in α-space. Although f is an approximate model, the intrinsic errors it carries only affects the coordinate values of the periodic orbits in this bifurcation diagram. The construction of a bifurcation diagram on the dynamics of the map f itself produces an identical but shifted image of the bifurcation diagram presented in Fig. 8, in the parameter space.

5 Algorithms for studying the 1-D map f

In this chapter we present some numerical algorithms to implement the approximate 1-D map f, to iterate this Poincaré map, to locate its fixed points and to compute the corresponding Schwarzian derivative's graph. These algorithms are introduced in order to provide a complete set of tools to analyse f and then derive some qualitative predictions on the dynamics of (1).

Here we keep most of notation introduced in Reference 4. The vector field associated with equation (1) is odd-symmetric and affine in the three regions D_{-1}, D_0 and D_1 partitioned by the $U_{-1} = \{(x,y,z)/x = -1\}$ and $U_1 = \{(x,y,z)/x = 1\}$ symmetric planes. This piecewise-linear vector field is characterized by the six parameter family $\{\tilde{\sigma_0}, \tilde{\omega_0}, \tilde{\gamma_0}, \tilde{\sigma_1}, \tilde{\omega_1}, \tilde{\gamma_1}\}$, where $(\tilde{\sigma_0} \pm i\tilde{\omega_0}, \tilde{\gamma_0})$ and $(\tilde{\sigma_1} \pm i\tilde{\omega_1}, \tilde{\gamma_1})$ are the eigenvalues of (1) in D_0 and D_1, respectively. In order to simplify analysis, affine changes of coordinates reduce each of the three linear vector fields to a canonical real Jordan form. In the text, "transformed D_0" will refer to the region D_0 with the new coordinate system.

5.1 Derivation of the 1-D map

The construction of f is entirely symmetric with the construction of the 1-D map π^* described in Reference 4. The main contribution of this approach is the derivation of explicit formulas to compute the coordinates of points belonging to the graph of f. The algorithm presented here is an extension of the algorithm established in Reference 9 :

Step 1 Given the parameters $(\alpha, \beta, m_0, m_1)$, calculate the eigenvalue family $\{\tilde{\sigma_0}, \tilde{\omega_0}, \tilde{\gamma_0}, \tilde{\sigma_1}, \tilde{\omega_1}, \tilde{\gamma_1}\}$, using Laguerre's or Cordano's method to solve the two third order characteristic equations associated with (1).

Step 2 Normalize the eigenvalues via $\gamma_0 = \tilde{\gamma_0}/\tilde{\omega_0}$, $\sigma_0 = \tilde{\sigma_0}/\tilde{\omega_0}$, $\gamma_1 = \tilde{\gamma_1}/\tilde{\omega_1}$, $\sigma_1 = \tilde{\sigma_1}/\tilde{\omega_1}$ and $k = -\tilde{\gamma_0}/\tilde{\gamma_1}$.

Step 3 Find the coordinates of the fundamental points A_0, B_0 in the transformed D_0 region via (2.20), (2.21) and (2.22) of Reference 4. Calculate the affine connection map Φ using the explicit formulas (4.40) and (4.44) of Reference 4.

Step 4 Given two return times t_0 and t_0', $0 \leq t_0, t_0' < \infty$, calculate the associated inverse-return time functions :

$$u^+(1, t_0) = \frac{(\varphi_0^{t_0}(B_0), h) - 1}{(\varphi_0^{t_0}(B_0 - A_0), h)} \quad \text{and} \quad u^-(1, t_0') = \frac{(\varphi_0^{t_0}(B_0), h) + 1}{(\varphi_0^{t_0}(B_0 - A_0), h)}$$

respectively. Where $h = (1, 0, 1)^T$ denotes the normal vector from the origin to V_1 in the transformed D_0 region, and $(.,.)$ the usual scalar product in R^3.
Repeat this operation with $t_0 = t_0 + \Delta t$ ($t_0' = t_0' + \Delta t$) until t_0 (t_0') is a first return time.

Step 5 From t_0 and the value of u obtained in step 4, calculate $\mathbf{x}(u) = A_0 u + (1 - u) B_0$ and the intersecting point $\mathbf{y}(u)$ [1] of the trajectory starting from $\mathbf{x}(u)$ with V_1, in the transformed D_0 region :

$$\begin{aligned} \mathbf{y}(u) &= \varphi_0^{t_0}(\mathbf{x}(u)) \\ &= (e^{\sigma_0 t_0}(x_x \cos t_0 - x_y \sin t_0), e^{\sigma_0 t_0}(x_x \sin t_0 + x_y \cos t_0)) \end{aligned}$$

Step 6 Convert the coordinates of points $\mathbf{x}(u)$ and $\mathbf{y}(u)$ from the transformed D_0 region to the transformed D_1 region via the connection map Φ.

Step 7 Calculate $\mathbf{X}(u)$, the intersection of the trajectory starting from $\mathbf{x}(u)$ and spiraling inward (in backward time) with the Poincaré line (the line $\overline{ON_{1\infty}}$ in Reference 4).

$$\mathbf{X}(u) = (1 + x_y^2)^{\frac{1}{2}} exp[-\sigma_1(\pi + \arctan x_y)]$$

[1] Here the coordinates of $\mathbf{x}(u)$ and $\mathbf{y}(u)$ are denoted by (x_x, x_y, x_z) and (y_x, y_y, y_z) respectively.

Step 8 Calculate $\mathbf{Y}(u)$, the abscissa on $\overline{ON_{1\infty}}$ of the intersection of the trajectory starting from $\mathbf{y}(u)$ with the Poincaré plane (the plane W_1 in Reference 4).
If the trajectory emanating from (y_x, y_y) loops once in D_1 before entering D_0, the calculation of $\mathbf{Y}(u)$ necessarily requires an analytic algorithm. Considering that the main contribution of this algorithm is to derive the *exact* representation of the graph of f, we recommend a global analytical approach (Runge-Kutta method for example) in such cases. Stop the simulation.
Else $\mathbf{Y}(u)$ is given by the following explicit equations :

$$\mathbf{Y}(u) = (y_x^2 + y_y^2)^{\frac{1}{2}} exp[-\sigma_1 \arctan \tfrac{y_y}{y_x}] \qquad \text{if } y_x \leq 0$$
$$\mathbf{Y}(u) = (y_x^2 + y_y^2)^{\frac{1}{2}} exp[-\sigma_1(\pi - \arctan \tfrac{y_y}{y_x})] \quad \text{if } y_x > 0$$

Step 9 $(\mathbf{X}(u), \mathbf{Y}(u))$ defines a point of the 1-D map f. Repeating the algorithm from step 4 to step 8, adding Δt to t_0 at each loop, a series of data points $\{X_p^1, Y_p^1\}_p$ belonging to the graph of the 1-D map f is obtained.

In the above algorithm, the 1-D map f is constructed without solving any transcendental equation and is a piecewise-smooth function[4]. We refer the reader to other remarks of interest on this algorithm in Reference[9].

5.2 Practical algorithms for studying the 1-D map f

Since a major interest of the approximate Poincaré map is to find the periodic orbits of (1), the derivation of an iterated map is of great interest. The following algorithm performs the iteration of f on the basis of the series of data points $\{X_p^1, Y_p^1\}_p$ obtained with the previous algorithm. Here the Poincaré map to be iterated is considered as a piecewise-affine function (the linear interpolation of the points $\{X_p^1, Y_p^1\}$) :

Step 1 Acquisition of the number of iterations (n).

Step 2 Given an abscissa x_0, determine the index j such that :

$$X_j^1 \leq x_0 \leq X_{j+1}^1$$

Step 3 Calculate the corresponding ordinate y_0 on the graph of f :

$$y_0 = f(x_0)) = \frac{(Y_{i+1}^1 - Y_i^1)x_0 - (Y_{i+1}^1 X_{i^1} - Y_i^1 Y_{i+1}^1)}{X_{i+1}^1 - X_i^1}$$

Step 4 Repeat n times the algorithm from step 2 to step 3 with $x_0 = y_0$.

Step 5 (x_0, y_0) defines a point of the 1-D map f^n. Repeating the algorithm from step 2 to step 4, adding Δx to x_0 at each loop, a series of data points $\{X_p^n, Y_p^n\}_p$ belonging to the graph of f^n is derived.

In order to find the periodic orbits of the 1-D map $f^n (n \geq 1)$, calculate the distance $d_p^n = (Y_p^n - X_p^n)$. Each time its sign changes $(d_p^n d_{p+1}^n < 0)$, the 1-D map f^n has a fixed point between (X_p^n, Y_p^n) and (X_{p+1}^n, Y_{p+1}^n). Then interpolate linearly the coordinates (x, x) and the slope of f^n at this point from the formula :

$$\begin{array}{lcl} x & = & (Y_{p+1}^n X_p^n - Y_p^n X_{p+1}^n)/((Y_{p+1}^n - Y_p^n) - (X_{p+1}^n - X_p^n)) \\ d(f^n(x))/dx & \simeq & (Y_{p+1}^n - Y_p^n)(X_{p+1}^n - X_p^n) \end{array}$$

Using the inverse linear transformation[4], transform this fixed point back into the original coordinate system. Then it is possible to look for a corresponding periodic orbit in the 3-D system.

In some control parameter intervals, the 1-D Poincaré map of the nonlinear system (1) is *unimodal.* Geometricaly, this occurs when the image of the first maximum of f, $f(x_{max_1})$, is less than the abscissa of the first minimum x_{min_1}. In such a case, the interval $[0, x_{min_1}]$ is invariant for f : $f([0, x_{min_1}]) \subset [0, x_{min_1}]$. Considering that most of theorems established so far on one-dimensional unimodal maps f require the Schwarzian derivative of f to be negative, a numerical computation of the Schwarzian derivative's graph of f provides usefull information. The Schwarzian derivative of f at x, denoted $Sf(x)$, is defined by :

$$Sf(x) = \frac{f'''(x)}{f'(x)} - \frac{3}{2}\left(\frac{f''(x)}{f'(x)}\right)^2$$

Then, given a point (X_p^1, Y_p^1) on the 1-D Poincaré map's graph, calculate $Sf(X_p^1)$ by using the following approximations for f', f'' and f''' :

$$\begin{cases} f'(X_p^1) = \frac{Y_{p+1}^1 - Y_{p-1}^1}{X_{p+1}^1 - X_{p-1}^1} \\ f''(X_p^1) = 2\left[\frac{Y_{p+1}^1 + Y_{p-1}^1 - f'(X_p^1)(2X_p^1 - X_{p+1}^1 - X_{p-1}^1)}{(X_{p+1}^1 - X_p^1)^2 + (X_p^1 - X_{p-1}^1)^2}\right] \\ f'''(X_p^1) = 6\left[\frac{Y_{p+2}^1 - Y_{p-2}^1 - f'(X_p^1)(X_{p+2}^1 - X_{p-2}^1) - \frac{1}{2}f''(X_p^1)\left((X_{p+2}^1 - X_p^1)^2 - (X_p^1 - X_{p-2}^1)^2\right)}{(X_{p+2}^1 - X_p^1)^3 + (X_p^1 - X_{p-2}^1)^3}\right] \end{cases}$$

For almost all parameter values of the Chua's equations (1), the Schwarzian derivative is found to be positive in some small intervals of $[0, b]$. However, it still remains possible to find invariant and unimodal intervals on the graph of f in which the Schwarzian derivative is negative. But here, no general rules apply.

6 Conclusion

Though the 1-D map f is an approximate model of Chua's circuit, it provides valuable qualitative and qualitative results on the periodic orbits of the original model (1). An in-depth study of f demonstrate the consistancy of our one-dimensionnal model.

We conjecture that the topography of the periodic orbits of the 1-D map in the parameter space is identical to the topography of periodic orbits in the 3-D dynamical system (1), but infinitesimaly shifted. This shift leads to discrepancies between the approximate and the original model periodic orbits, most generally detected when the period order is high.

Appendix A : Correspondence between the periodic points of the 1-D Poincare map f and the periodic orbits of the dynamical system (1)

Here, the 2-D Poincaré map P we use to analyse the dynamics of the flow (1) is the original 2-D map from which we derived the approximate map f in Section 2. Its domain in the outer region D_1 with the appropriate coordinate system[4] which reduces the vector field to the real Jordan form is the plane $W_1 = \{(x, y, z) : y = 0\}$. Fig. 10 shows the

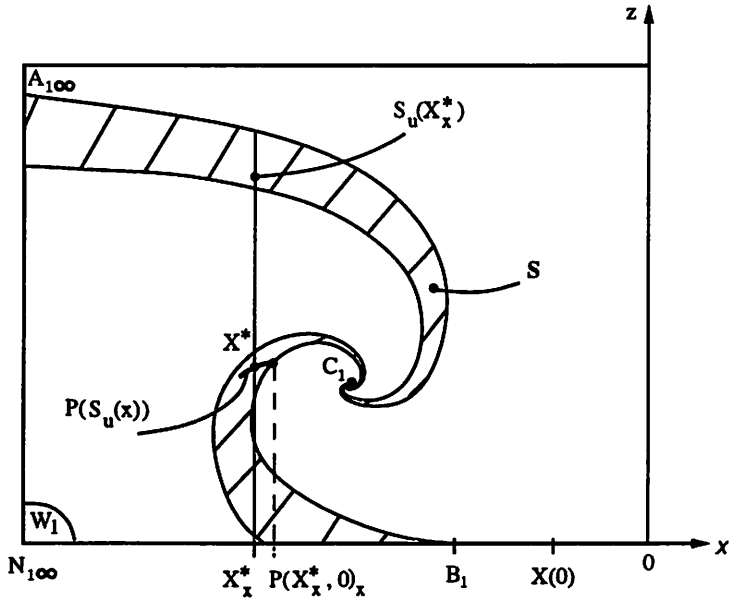


Figure 10: The 2-D Poincaré map P defined on the plane W_1. X^* is a fixed point of P. The double-snake shadowed area (S) has been intentionaly enlarged.

image of the region $\{(x, z) : x \leq X(0)\}$ through the mapping P ($X(0)$ is the limit beyond which a point on W_1 crosses or not a boundary plane before beeing mapped on W_1 by P). On this graph the semi-infinite line $\overline{ON_{1\infty}}$ represents the intersection of W_1 whith the 2-D unstable manifold in the redion D_1. The image of $\overline{X(0)N_{1\infty}}$ through the mapping P is the double spiral $B_1C_1A_{1\infty}$. Following the concepts presented in Section 2, the approximate 1-D map f is constructed on the semi-infinite line $\overline{ON_{1\infty}}$: the image of a point $(x, 0)$ based on $\overline{X(0)N_{1\infty}}$, mapped by P on the union of the double spiral with the segment $\overline{OB_1}$, through the flow f is the projection of $P(x, 0)$ on the semi-infinite line $\overline{ON_{1\infty}}$. The rigorous definition of f is stated as follow :

$$f : \overline{ON_{1\infty}} \longrightarrow \overline{ON_{1\infty}}$$
$$f(M) = P(M) \mid_x$$

Since we use an approximate model to predict the behavior of the 3-D system, it is essential for the reliability of any result derived from the model to have a good knowledge

of the *intrinsic* errors it carries, independently of the external errors added to the model through computer simulations. Here we restrict our attention to the periodic orbits of both 1-D and 2-D Poincaré maps, and particularly to the period-1 orbits. The same reasoning as follow applies to limit cycles of higher period. We try to provide an answer to the following question :

> Is there a general equivalence relationship between the period-1 orbits of the 1-D map and the period-1 orbits of the 3-D system (1)?

First let us consider a fixed point X^* of the 2-D Poincaré map P. Due to the attraction towards the unstable manifold in D_1, the image of the segment $S_u(X_x^*)$ (Fig. 10) on the boundary plane U_1 under the flow is confined to a very small neighborhood U of $\mathbf{x} = \{\varphi_1^t(X_x^*, 0)\}_{t>0} \bigcap U_1$. This neighborhood is then returned by the flow in the central region on the same boundary plane U_1 : a small area U' (in case the flow intersects the lower boundary plane U_{-1} we apply the invariant flip map $\mathbf{x} \longrightarrow -\mathbf{x}$ to the intersecting orbits). Eventualy, the flow projects U' in an infinitesimal area V centered in X^*. Since the vector field is continuous, the image of $S_u(X_x^*)$ segment under the 2-D Poincaré map P is necessarily a continuous curve. This curve included in V of course intersects the points X^* and $P(X_x^*, 0)$ and lies in S (S is the "double-snake" shadowed area in Fig. 10).

In f approximation, under the asumption that the double-snake area on W_1 is squeezed into a thin line sitting infinitesimaly close to $\overline{ON_{1\infty}}$, we consider that only the x coordinate is relevant to the dynamics of P. Then the approximate 1-D map is constructed on the semi-infinite line $\overline{ON_{1\infty}}$. So in order to obtain an exact characterization of a limit cycle of the 1-D map, we should at least have the following equality : $P(X_x^*, 0)_x = X_x^*$. But there is no evidence for $P(X_x^*, 0)_x$ to be strictly equal to X_x^*. Rigorously, to obtain the equality, one of the two following properties would have to be fulfilled :

$$P[S_u(X_x^*)] \text{ is a straight vertical segment } (x \text{ constant}), \tag{6}$$

$$X_x^* \text{ belongs to the double spiral } B_1C_1A_{1\infty}. \tag{7}$$

We assume that the first assertion is almost never true. And concerning the latter, it remains impossible, as far as we know, to derive a general property till the attractor of Chua's cicuit does not lie on the double spiral $B_1C_1A_{1\infty}$. See Reference 4 for a detailed analysis of the geometric structure of attractors exhibited in (1).

Now let us consider a fixed point, $\mathbf{x}$ $\left(\in \overline{ON_{1\infty}} \right)$ of the 1-D Poincaré map. Then $f(\mathbf{x}) = \mathbf{x}$. This means that P maps $\mathbf{x}$ into a point X lying on the double spiral $B_1C_1A_{1\infty}$, and which coordinates fill the equality :

$$X_x = \mathbf{x}$$

Using the same reasoning as above, we come to the conclusion that at least one of the assertions (6) and (7) have to be true for X to be a limit cycle of the 3-D system (1).

Obviously there is not any rigorous general equivalence relationship between the period-1 orbits of the 1-D map f and the period-1 orbits of the 2-D map P. However, the *existence* of a fixed point in the approximated 1-D map f can still condition the existence of a periodic orbit in the 3-D system with slightly different characteristics (stability, initial conditions, ...), and vice-versa. In the simulations performed so far, the numerical differences between the two systems appear to be negligible for the periodic orbits of low order. As a general rule the periodic orbits seem to be *shifted* in the parameter space. Each time an irregularity is observed, the shift between the periodic windows forbids any overlap. In such a case, no periodic orbit can be observed at the same time in both the approximate and original dynamical systems.

As well as predicting the existence of the periodic orbits of Chua's circuit, an another interresting feature of the 1-D map f is its ability to predict the stability of these periodic orbits. The characteristic multiplier of a periodic orbit of f (the slope of the graph of f at the fixed point for a period-1 orbit) is almost equal to the greatest characteristic multiplier (different from unity) of the corresponding periodic trajectory in the 3-D system. An explanation of this property can be derived from the study of the 2-D poincare map P. Defining P as follows :

$$\begin{array}{rcl} P : W_1 & \longrightarrow & W_1 \\ (x,z) & \longrightarrow & (P_1(x,z), P_2(x,z)) \end{array}$$

the stability of an hyperbolic fixed point X^* of P is determined by the eigenvalues of the linearized map $DP(X^*)$. Using Floquet theory[8], these eigenvalues are known to be equal to two of the three charateristic multipliers of the periodic trajectory. The other multiplier, associated with pertubations along the periodic trajectory, is always equal to unity. The linearized map is written as :

$$DP(x,z) = \begin{pmatrix} \frac{\partial P_1}{\partial x}(x,z) & \frac{\partial P_1}{\partial z}(x,z) \\ \frac{\partial P_2}{\partial x}(x,z) & \frac{\partial P_2}{\partial z}(x,z) \end{pmatrix}$$

Due to the contraction of the flow toward $\{(x,y,z)/z = 0\}$, the variations of P along z-axis can be considered almost equal to zero. Therefore the linearized map is rewriten as follow:

$$DP(x,z) \simeq \begin{pmatrix} \frac{\partial P_1}{\partial x}(x,z) & 0 \\ \frac{\partial P_2}{\partial x}(x,z) & 0 \end{pmatrix}$$

Then the eigenvalues assotiated with the fixed point X^* of P are almost equal to $\frac{\partial P_1}{\partial x}(X_x^*, X_z^*)$ and 0. Therefore, as a matter of fact, any periodic orbit of (1.1) has a stable manifold whose dimension is *at least* equal to 1, due to the zero value of one of the characteristic multipliers.

By construction, we have the following relationship between f and P :

$$f(x) = P_1(x,0)$$

Then the charateristic exponent of a periodic point of the one-dimensional map is equal to $\frac{\partial P_1}{\partial x}(x,0)$. We conclude that the slope of the one-dimensional Poincaré map at a fixed point (the product of the slope at the sequence points for period-n orbits, with $n > 1$) is a good approximation of the greatest characteristic multiplier ($\frac{\partial P_1}{\partial x}(x,z) \simeq \frac{\partial P_1}{\partial x}(x,0)$), different from unity, of the periodic orbit.

Appendix B : Experimental results

			Initial conditions			
	α	T	X_0	Y_0	Z_0	Comments
Π_0^1	7	2.085	1.81217	0.31545	-1.5214	(stable)
	7.5	2.172	2.10315	0.0899	-3.0072	(stable)
	8	2.25	2.10455	0.25515	-2.49746	(stable)
	8.1629	2.255	1.65234	-0.20943	-2.7124	Period doubling bif.
	8.3	2.3	1.65	-0.21	2.7	
	10.5	2.75				
	11.6926	3.45	1.47245	0.16502	-1.02651	(stable)
	10.915409923	4.582	1.00	-0.29859	-1.44218	(stable)
	11.05	5.055	1.69325	-0.102	-2.57302	
	11.136	5.602	1.71205	-0.092	-2.57302	
Π^1	8	2.287	2.954	1.27713	-3.07322	
	9	2.482	2.08	-0.36775	-4.51385	
	10	2.702	-2.3053	-0.0934	3.70658	
	11	2.968	2.03315	0.1	-3.07322	
	12	3.28	1.65241	-0.12841	-2.67183	
	13.736	4.525	-1.53955	0.11236	2.2777	Saddle node bif.
	13.715	4.615	1.93349	0.15572	-1.96968	(stable)
	13.706	4.7	1.92344	1.15615	-1.9428	Pitchwork bif.
	13	5.257	2.0541	0.14884	-1.86609	
	10.4	6.657	2.05482	0.21377	-2.23035	
	11.2	8.747	2.16255	0.16592	-1.70372	

Table 1: The continuous deformation of Π_0^1 and Π^1

Table 1 gives the coordinates of some periodic orbits on the (α, T) plane, used to interpolate the bifurcation diagram of Chua's circuit presented in Section 4. For each point, initial conditions based on the corresponding stable periodic trajectory are provided. Rigorously, these initial conditions are included in an infinitesimal neighborhood of the corresponding periodic trajectory, but not exactly on the trajectory itself.

All our values have been obtained using INSITE, a software toolkit for the analysis of nonlinear dynamical systems.

Appendix C : CHUA 1-D Map User's Guide

CHUA 1-D Map program runs on PC, on top of both the graphical user interface toolkit *Menuet* and MetaWindow Software Corporation's MetaWINDOW graphics product.

After starting the program, the user has access to any of the following functions by clicking with the mouse on the appropriate menu button.

Parameters

Choose the values of Chua's circuit parameters α, β, m_0 and m_1. To set the value of a particular parameter, choose the **Parameter** option in the main menu and then select the parameter to be modified by clicking on the appropriate dialogue box. Then enter the parameter value by typing the desired number. Table 2 shows the default choices of parameter values.

α	9.0
β	100/7
m_0	-8/7
m_1	-5/7

Table 2: Default parameters for dimensionless Chua's circuit.

Options

The user is prompted to enter the number of data points he wants to use to interpolate the graphs of the one-dimensional map and its iteration. Here, he can also specify the maximum number of iteration of the 1-D map f and the value of the return time step (see Section 5). To enter new values, type the desired number in the dialog box. Table 3 shows the default choices of the option values.

Return time step	0.02
Max. number of iteration	8
Data point number for **Plot**	1000
Data point number for **Iterate**	1000

Table 3: Default option values.

Plot

Display the graph of the aproximate one-dimensionnal Poincaré map f of Chua's circuit and the graph of the Schwarzian derivative sign.

Iterate

The user is first prompted to enter a number of iteration i. Then the **Iterate** function plot the graph of the i-th iterate of the 1-D map f.

Eigenvalues

The **Eigenvalues** function displays the five normalized eigenvalue parameters of Chua's circuit, corresponding to the current set of parameters $(\alpha, \beta, m_0, m_1)$.

Min/Max

The **Min/Max** function stores the coordinates of minimum and maximum values of the 1-D map f in the file *minmax.dat.* The abscissa of the critical point a on the graph of f is also stored. Data files include the parameters for the simulation as a header and are written in ASCII format.

Fixed Point

The **Fixed Point** function localizes the fixed points of the latter plot of the 1-D map (either the map f or its iterates). For each fixed point, the function returns the slope of the map at this point and initial conditions based on the corresponding periodic orbit in the original 3-D dynamical system. All the results are stored in the file *fixpt.dat.* Data files include the parameters for the simulation as a header and are written in ASCII format.

Save

The **Save** function prompts the user for the names of files in which to save data from the latter simulation. Data files include the parameters for the simulation as a header and are written in ASCII format. The file does not include the Schwarzian derivative sign simulation.

Quit

The **Quit** button exits the program gracefully.

Appendix D : C- Code routines for the construction of the 1-D map f.

```
/*****************************************************************************/
/**                                                                         **/
/**   This program is designed to study an approximate one-dimensionnal map **/
/**   of Chua's circuit : an original 2-D poincare map is reduced to a map on **/
/**   on a line into itself under the standing assumption that the flow is  **/
/**   contracted into an infinitesimal neighborhood of the unstable manifold **/
/**   of the outer regions.                                                 **/
/**   The following functions compute :                                     **/
/**       -the derivation of this 1D-map with explicit equations            **/
/**       -the calculation of the associated Schwarzian derivative's graph  **/
/**       -the iterations of this 1-D map                                   **/
/**       -the fixed points of any iteration of the 1-D map                 **/
/**                                                                         **/
/**                    Written by Marc Genot 05/20/1992                     **/
/**                   Copyright University of California                    **/
/**                        Version 1 - 05/20/1992                           **/
/**                                                                         **/
/*****************************************************************************/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "complex.c"
#include "util.c"

#define YES 1
#define NO 0
#define Dim 3              /** dimension of the peacewise-linear circuit **/
#define Tmax 10            /** maximum return time **/
#define pi 3.14159265
#define MAX_ITER 8         /** maximal iteration value of the 1D poincare map **/


typedef struct{
         double x,y;
         }point_2D;
point_2D A0,B0;              /** Fundamental points A and B in the D0 transformed un
it **/
long int M,IT_POINT;
double dT;
double PHY[2][2];            /** affine connection map **/
double alpha,beta,m0,m1;     /** dimensionless parameters of the Chua's circuit **/
double sigma0,sigma1,gamma0,gamma1,k ; /** Normalized eigenvalues parameters **/
double X1min,X1max,X2min,X2max;  /** minimum and maximum values of X(u) for u=u+ and
 u=u- **/


void EIGENVALUES();
void INITIALISATIONS();
void MAP();
void FIND_MINMAX();
void ITERATION();
void SCHWARZIAN_DER();
void SEARCH_FIXE_POINTS();


main(int argc, char *argv[])
{
     static char s10[] = { "poincare" }; /** name of the file's storage of points be
longing to the 1D map **/
     static char s20[] = { "/net/ataraxia/bin/xgraph " };  /** UNIX command line to
execute xgraph **/
     static char s1[10],s2[50];
     int it,visu,schwcal,sfp,itr,simu,minmax,interpol;
     long nb;
     double **X,**Xit,Xa,X0;
     void yes_or_no();

     if(argc != 3) {
          printf("Usage: run <Nb of data points which define the map f>\n");
          printf("             <Nb of data points which define the iterate maps>\n
);
          printf("Note : In case of a malfunction of the graphics, check the\n");
          printf("       command line running 'xgraph' in the main routine.\n");
          exit(0);
          }
     M = atol(argv[1]);
     IT_POINT = atol(argv[2]);
     dT = 2.0*((double)Tmax)/((double) M);
     X = dmatrix(0,1,0,M-1);
     Xit = dmatrix(0,1,0,IT_POINT-1);
     printf("\n");
     printf("Display the maximun/minimum values of the 1D_map?"); yes_or_no(&minmax
;
     printf("Calculate the schwarzian lie's sign?....."); yes_or_no(&schwca
);
     printf("Display graphs?................................."); yes_or_no(&visu);
     if (visu==YES){
          printf("Interpolate/print ( y = x ) line?.............");
          yes_or_no(&interpol);
        }

     do{
          EIGENVALUES();
          INITIALISATIONS();
          strcpy(s1,s10);
          MAP(s1,X,&Xa,&X0,&nb);
          if(minmax==YES) FIND_MINMAX(X,Xa);
          if(visu==YES){
               strcpy(s2,s20);
               strcat(s2,s1);
               if(interpol==YES) strcat(s2,"  -t 1D_poincare_map &");
               else strcat(s2," -nl -p -t 1D_poincare_map &");
               system(s2);
               }

          if(schwcal==YES){
               SCHWARZIAN_DER("lie",X,nb);
               if(visu==YES){
                    strcpy(s2,s20);
                    strcat(s2,"lie -nl -P -t Schwarzian_lie &");
                    system(s2);
                    }
             }
          printf("Print out fixed points?........................."); yes_or_no(&s
p);
          if(sfp==YES) SEARCH_FIXE_POINTS(X,Xit,Xa,X0,nb,1);
          printf("Iterate the 1D poincare map?....................."); yes_or_no(&i
r);
          if(itr==YES){
               do{
                    strcpy(s1,s10);
                    strcat(s1,"_i");
                    ITERATION(s1,X,Xit,&it);
                    if((visu==YES)&&(it!=1)){
                         strcpy(s2,s20);
                         strcat(s2,s1);
                         if(interpol==YES) strcat(s2," -t iterated_poincare_map &")
```

```
                    else strcat(s2," -nl -p -t iterated_poincare_map &");
                    system(s2);
                    }
               if(it!=1){
                    printf("  Print out fixed points?....."); yes_or_no(&sfp);
                    if(sfp==YES) SEARCH_FIXE_POINTS(X,Xit,Xa,X0,nb,it);
          }
               printf("  Continue iteration?........"); yes_or_no(&itr);
               }while (itr==YES);
       }
         printf("Continue simulation with other parameters?.......");yes_or_no(&sim
u);
         printf("\n");
         }while(simu==YES);

     free_dmatrix(X,0,1,0,M-1);
     free_dmatrix(Xit,0,1,0,IT_POINT);

     printf("          1D poincare map in 'poincare' file.\n");
     printf("          Last iteration of the 1D poincare map in 'poincare_i' file.\n
");
     printf("          Schwarzian lie's sign in 'lie' file.\n\n");
}


/** Return Yes or NO and store the answer in 'reply' adress. **/
/** By default the answer is NO.                             **/
void yes_or_no(reply)
int *reply;
{
     char re;

     printf(" [y or n] : ");
     scanf("%s",&re);
     if (re=='y') *reply = YES;
     else *reply = NO;
}


/** Acquisition of the dimensionless parameters of the Chua's circuit **/
/** Calculation of the associated normalized parameter eigenvalues    **/
/** using the Laguerre's method.                                      **/
void EIGENVALUES()
{
     int i,j0,j1,polish;
     double m,re0[Dim],im0[Dim],re1[Dim],im1[Dim],p0,p1;
     fcomplex a[Dim+1], roots0[Dim+1], roots1[Dim+1];
     void zroots();

     printf("##########################################################################\
n");
     printf("##                Enter the dimensionless parameters                    ##\
n");
     printf("##                          for Chua's cicuit                           ##\
n");
     printf("##########################################################################\
n");
     printf("                         alpha = ");
     scanf("%lf",&alpha);
     printf("                         beta  = ");
     scanf("%lf",&beta);
     printf("                         m0    = ");
     scanf("%lf",&m);
     m0 = m + 1;
     printf("                         m1    = ");
     scanf("%lf",&m);
     m1 = m + 1;
     polish = 1;
     for (j0=0;j0<=1;j0++){
          m = (1.-j0)*m0+j0*m1;
          a[0].r = alpha*beta*m;
          a[1].r = alpha*m+beta-alpha;
          a[2].r = alpha*m+1.;
          a[3].r = 1.;
          for(i=0;i<=Dim;i++)  a[i].i=0.;
          if (j0==0) zroots(a,Dim,roots0,polish);
          else zroots(a,Dim,roots1,polish);
          }

     for (i=1;i<=3;i++){
          if (roots0[i].i==0.){
               p0 = roots0[i].r;
               j0 = (i+1)%3;
        }
          if (roots1[i].i==0.) {
               p1 = roots1[i].r;
               j1 = (i+1)%3;
        }
          }

     /** normalized eigenvalue parameters **/
     gamma0 = p0/fabs(roots0[j0].i);
     sigma0 = roots0[j0].r/fabs(roots0[j0].i);
     gamma1 = p1/fabs(roots1[j1].i);
     sigma1 = roots1[j1].r/fabs(roots1[j1].i);
     k = -p0/p1;
     printf("\n  (gamma0, sigma0, gamma1, sigma1 ,k)=\n");
     printf("      (%7.4f,%7.4f,%7.4f,%7.4f,%7.4f)\n\n",gamma0,sigma0,gamma1,sigma1,
);
}


/** Calculate the explicite value of the affine connextion map and the fundamental
*/
/** points' coordinates A0 and B0 **/
void INITIALISATIONS()
{
     double a,p0,Q0,Q1;

     p0 = sigma0+(k*(sigma0*sigma0+1.0)/gamma0);
     Q0 = (sigma0-gamma0)*(sigma0-gamma0)+1.0;
     Q1 = (sigma1-gamma1)*(sigma1-gamma1)+1.0;
     a = (sigma1*sigma1+1.0)/(k*(sigma0*sigma0+1)*(k+1.0)*Q1*gamma1);

     /** calcul of the fondamental points' coordinates **/
     A0.x = 1.0;
     A0.y = p0;
     B0.x = gamma0*(gamma0-sigma0-p0)/Q0;
     B0.y = gamma0*(1.0-p0*(sigma0-gamma0))/Q0;

     /** expression of the affine connextion map **/
     PHY[0][0] = -a*gamma1*(k+1.0)*(Q0+gamma0*(sigma0-gamma0)*((1.0/k)+1.0));
     PHY[0][1] = a*gamma0*gamma1*(k+1.0)*((1.0/k)+1.0);
     PHY[1][0] = -a*gamma0*((1.0/k)+1.0)*(sigma0-gamma0)*(sigma1*(sigma1-gamma1)+1.
)
                 -a*gamma1*(k+1.0)*(sigma1-gamma1)*(sigma0*(sigma0-gamma0)+1.0);
```

```
     PHY[1][1] = a*gamma0*((1.0/k)+1.0)*(Q1+gamma1*(sigma1-gamma1)*(k+1.0));
}



/** Derivation of the 1D Poincare map. The points belonging to its graph are **/
/** stored in the array (2,M) X, and written in fname file.                   **/
void MAP(fname,X,xa,X0,nb)
char *fname;
double **X,*xa,*X0;
long *nb;
{
     long i;
     int double_loop = NO;
     int monot_1,monot_2 = YES;
     long p = 1;
     double t0 = 0.0;
     double umax = 0.0;
     double umin = 100.0;
     double u,IRT_f(),**Xbuff;
     void xy_calcul(),print_map();
     FILE *fp;


     Xbuff = dmatrix(0,1,0,M-1);
     X[0][0] = 0.; X[1][0] = 0.;
     *X0 = sqrt(1+sigma1*sigma1)*exp(-sigma1*(pi+atan(sigma1)));
     while (t0<Tmax){
          do{
               t0 += dT;
               u = IRT_f(t0,1);              /** evaluation of u+(1,t0) **/
               if (t0>Tmax) break;
               if(u<=umax) monot_1 = NO;
        }while ((u<=umax) && (u<=1));  /** evaluation of the first return time t0 **
/
          if (u>1) break;
          xy_calcul(&X[0][p],&X[1][p],u,t0,*X0,1);
          umax = u;
          p++;
          }
     p--;
     X1max=X[0][p];

     *nb=p+1;
     t0 = 0.5; p = 0;
     while (t0<Tmax){
          do{
               t0 += dT;
               u = IRT_f(t0,-1);             /** evaluation of u_(1,t0) **/
               if (t0>Tmax) break;
               if(u>=umin) monot_2 = NO;
        }while ((u>=umin) && (u>=1)); /** evaluation of the first return time t0 **/

          if (u<1) break;
          xy_calcul(&Xbuff[0][p],&Xbuff[1][p],u,t0,*X0,-1);
          if (p==0) X2max=Xbuff[0][p];
          umin = u;
          p++;
          }
     *xa = (X1max+Xbuff[0][p-1])/2.0;

     for (i=0;i<=p-1;i++){
          X[0][*nb+i] = Xbuff[0][p-i-1];
          X[1][*nb+i] = Xbuff[1][p-i-1];
     }
     *nb = *nb+p;

     if(monot_1 == NO) printf("  WARNING : u+ is non monotonic\n");
     if(monot_2 == NO) printf("  WARNING : u- is non monotonic\n");
     print_map(fname,X,*nb);
     free_dmatrix(Xbuff,0,1,0,M-1);
}


/** Calculate the maximum and minimum values of the 1-D Poincare map. **/
/** The algorithm include a threatement of the discontinuities.       **/
void FIND_MINMAX(X,xa)
double **X,xa;
{
     double diff=0.0;
     long p;
     int sign,max_nb;

     p = 0 ; max_nb = 0; sign = 1;
     while(X[0][p]<=X1max){
          while(sign*diff>=0){
               p++;
               if (X[0][p]>X1max) break;
               diff=X[1][p]-X[1][p-1];
        }
          if (X[0][p]>X1max) break;
          if (fabs(diff)<=0.1){
               if (sign==1){
                    printf("  Coordinates of maximum %2d : Xu=%10.81f Yu=%10.81f\n"

                                   ++max_nb,X[0][p-1],X[1][p-1]);
             }
               else printf("  Coordinates of minimum %2d : Xu=%10.81f Yu=%10.81f\n"

                                   max_nb,X[0][p-1],X[1][p-1]);
               sign = -sign;
        }
          else diff=0.0;
          }
     printf("\n  Coordinates of 'a' (u = 1)    : Xa=%10.81f\n\n", xa);

     p++;
     diff = (X[1][p]-X[1][p-1]);
     if (diff>0){
          sign = 1;
          max_nb++;
          }
     else sign = -1;
     while(X[0][p]<X2max){
          while(sign*diff>=0){
               p++;
               if (X[0][p]>=X2max) break;
               diff=X[1][p]-X[1][p-1];
               }
          if (X[0][p]>=X2max) break;
          if (fabs(diff)<=0.1){
               if (sign==1){
                    printf("  Coordinates of maximum %2d : Xu=%10.81f Yu=%10.81f\n"

                                   ++max_nb,X[0][p-1],X[1][p-1]);
             }
               else printf("  Coordinates of minimum %2d : Xu=%10.81f Yu=%10.81f\n"
```

```
                        max_nb,X[0][p-1],X[1][p-1]);
                sign = -sign;
        }
          else diff=0.0;
   }
     printf("\n");
}


/** Acquisition of an iteration's order, it, and iteration the 1-D Poincare map. **/

/** The points belonging to its graph are stored in the array (2,M) Xit, and     **/

/** written in fname file.                                                       **/

void ITERATION(fname,X,Xit,it)
char *fname;
double **X,**Xit;
int *it;
{
     void iterate();

     *it = 0;
     printf("  How many iterations?..... : ");
     scanf("%d",it);
     if ((*it<=0) || (*it>MAX_ITER)){
          do{
               printf("  Value must be in [1..%d] ..... : ",MAX_ITER);
               scanf("%d",it);
        }while((*it<=0) || (*it>=MAX_ITER));
   }
     if (*it>1) iterate(*it,X,Xit,fname);
}



/** evaluation of the non-iterated poincare map's schwarzian lie,        **/
/** -1 and 1 values are returned in "name" for corresponding negative or **/
/** positive schwarzian lie respectively, assoiated whith X **/
void SCHWARZIAN_DER(name,X,nb)
char *name;                        /* file which will contains the schwarzian lie sign
 */
double **X;                    /* points which define the 1D poincare map */
long nb;                            /* number of points which define the 1D poincare m
ap */
{
     FILE *fp;
     double Xp[2][5];
     long j;
     int i,schwarzian_appr();

     fp = fopen(name,"w");
     for (j=1;j<=5;j++){
          Xp[0][j-1]=X[0][j];
          Xp[1][j-1]=X[1][j];
   }
     while(j<nb){
          if(fabs(Xp[1][4]-Xp[1][3])>=0.1){
               j += 3;
               for (i=0;i<=3;i++){
                    Xp[0][i] = X[0][j+i-4];
                    Xp[1][i] = X[1][j+i-4];
               }
               }
          else{
               fprintf(fp,"%lf %d\n",Xp[0][2],schwarzian_appr(Xp));
                    for (i=0;i<=3;i++){
                         Xp[0][i] = Xp[0][i+1];
                         Xp[1][i] = Xp[1][i+1];
                    }
               }
            Xp[0][4] = X[0][j]; Xp[1][4] = X[1][j];
            j++;
     }
       fclose(fp);
}



/** look for the fixed points of the (it)th iterated poincare map and    **/
/** print out the coresponding points in the real dynamical system whith **/
/** the slope of the poincare map for each fixed point                   **/
void SEARCH_FIXE_POINTS(X,Xit,Xa,X0,nb,it)
double **X,**Xit,Xa,X0;
long nb;
int it;
{
     double di,dj,ka,p1,ay,az,bz;
     long i;
     void analyse_fixe_point();

          ka = (m1-m0)/m1;
          p1 = sigma1+(sigma1*sigma1+1.0)/(gamma1*k);
          ay = (gamma0*gamma0+gamma0+beta-alpha*ka-(1.0-ka)*(gamma1*gamma1+gamma1+b
ta))/
                  (alpha*(gamma1-gamma0));
          az = (alpha*ka*gamma0+(1.0-ka)*gamma0*(gamma1*gamma1+gamma1+beta)
                  -gamma1*(gamma0*gamma0+gamma0+beta))/(alpha*(gamma1-gamma0));
          bz = -(gamma0*gamma0+gamma0+beta+alpha*gamma0*m0)/alpha;

     printf("#########################################################################
n");
     printf("##               Fixed point of the 1D poincare map                  ##
n");
     printf("##                          (%d-ith iteration)                        #
\n",it);
     printf("#########################################################################
n");
     printf("## X(0)=%10.8lf                                                       ##\n"
X0);
     printf("#########################################################################
n");
     if (it==1){
          di = X[0][1]-X[1][1];
          for(i=2;i<nb-1;i++){
               dj = X[0][i]-X[1][i];
               if((X[0][i]>=Xa)&&(X[0][i-1]<=Xa)){
          printf("## Xa
  ##\n");
             }
               if ((di*dj<=0)&&(fabs(X[0][i]-X[0][i-1])<=0.1)){
                    analyse_fixe_point(X[0][i-1],X[1][i-1],X[0][i],X[1][i],ka,p1,ay
az,bz);
             }
               di=dj;
          }
            }
```

```
     else{
          di = Xit[0][0]-Xit[1][0];
          for(i=1;i<IT_POINT-1;i++){
               dj = Xit[0][i]-Xit[1][i];
               if((Xit[0][i]>=Xa)&&(Xit[0][i-1]<=Xa)){
          printf("## Xa
  ##\n");
             }
               if ((di*dj<=0)&&(fabs(X[0][i]-X[0][i-1])<=0.1)){
                    analyse_fixe_point(Xit[0][i-1],Xit[1][i-1],Xit[0][i],
                                                    Xit[1][i],ka,p1,ay,az,bz);
      }
               di=dj;
        }
    }
     printf("##########################################################################\
n");
}


/** Calculation of the fixed point's parameters.  **/
void analyse_fixe_point(xi,yi,xj,yj,ka,p1,ay,az,bz)
double xi,yi,xj,yj,ka,p1,ay,az,bz;
{
     double slope,fix,z0,fp;

     fp = (yi*xj-yj*xi)/((yi-yj)-(xi-xj));
     fix = (yi*xj-yj*xi)/((yi-xi)-(yj-xj));
     z0 = -((az+ka)*sigma1-(bz+ka)*p1)*fix/(sigma1-p1)-ka;
     slope = (yi-yj)/(xi-xj);
     printf("## X(u)=%10.8lf Xo=(%8.6lf %8.6lf %8.6lf) Slope=%9.4f ##\n",
           fp,-(1.0-ka)*fix+ka,-(ay*sigma1-m0*p1)*fix/(sigma1-p1),z0,slope);
}


/** Iteration of the 1D_map **/
void iterate(iter,X,Xit,fname)
int iter;
double **X,**Xit;
char *fname;
{
     long p = 0;
     int i;
     double dX,Xc,Yc,which_x();
     FILE *fp;

     fp=fopen(fname,"w");
     fprintf(fp,"%16.14f %16.14f\n",0.,0.);
     fprintf(fp,"%16.14f %16.14f\n",X2max,X2max);
     fprintf(fp,"\n");
     fprintf(fp,"%16.14f %16.14f\n",0.,0.);
     dX = X2max/IT_POINT;
     Xc = dX;
     while (Xc<=X2max){
          Yc = which_x(Xc,X);                              /** Yc = f(Xc)          *
*/
          for (i=2;i<=iter;i++) Yc=which_x(Yc,X);          /** Yc = fofo..ofof(Xc) *
*/
          Xit[0][p] = Xc;
          Xit[1][p] = Yc;
          p++;
          if (Yc<=X2max)  fprintf(fp,"%16.14f %16.14f\n",Xc,Yc);
          else{
               fprintf(fp,"\n");
               do{
                    Xc += dX;
                    Yc = which_x(Xc,X);                        /** Yc = f(Xc)
**/
                    for (i=2;i<=iter;i++) Yc=which_x(Yc,X); /** Yc = fofo..ofof(Xc)
**/
                    Xit[0][p] = Xc;
                    Xit[1][p] = Yc;
                    p++;
                    }while((Yc>X2max) && (Xc<=X2max));
               }
          Xc += dX;
     }
     fclose(fp);
}


/** Find the two points in the 1-D Poincare map (X) such that xi is included **/
/** between the abscissa components of this two points.                      **/
double which_x(xi,X)
double xi,**X;
{
     long i = 0;
     double d,y;

     y = xi;
     if (xi<=X2max){
          do{
               d=xi-X[0][i++];
               }while(d>=0.0);
          i--;
          y = ((X[1][i]-X[1][i-1])*xi-X[1][i]*X[0][i-1]+X[1][i-1]*X[0][i])/
                              (X[0][i]-X[0][i-1]);
     }
     return(y);
}


/** Given a point Xp[][2] of the 1-D Poincare map, this function calculate  **/
/** the Schwarzian derivative. Only the sign of the derivative is returned. **/
int schwarzian_appr(Xp)
double (*Xp)[5];
{
     double f1,f2,f3,s,dT1,dT2,dT3,dT4;
     int sch;

     dT1 = Xp[0][3]-Xp[0][2];
     dT2 = Xp[0][2]-Xp[0][1];
     dT3 = Xp[0][4]-Xp[0][2];
     dT4 = Xp[0][2]-Xp[0][0];
     f1 = (Xp[1][3]-Xp[1][1])/(dT1+dT2);
     f2 = 2.0*(Xp[1][3]+Xp[1][1] - 2.0*Xp[1][2] + f1*(dT2-dT1))/(dT1*dT1+dT2*dT2);
     f3 = 6.0*(Xp[1][4]-Xp[1][0] + 0.5*f2*(dT4*dT4-dT3*dT3) - f1*(dT3+dT4))/
                              (dT3*dT3*dT3+dT4*dT4*dT4);
     s = f3*f1-1.5*f2*f2;
     if (s<=0) sch = -1;
     if (s>0) sch = 1;
     return(sch);
}


/** Calculate Xu and Yu corresponding to the values of u and t. **/
/** The parameter mode indicates weither u=u+ or u=u-.          **/
void xy_calcul(Xu,Yu,u,t,X0,mode)
```

```
double *Xu,*Yu,u,t,X0;
int mode;
{
     double p0,p1,r1,teta1,rX0;
     point_2D X,Y,X1,Y1;

     X.x = u*A0.x+(1.0-u)*B0.x;
     X.y = u*A0.y+(1.0-u)*B0.y;
     Y.x = mode*exp(sigma0*t)*(cos(t)*X.x-sin(t)*X.y);
     Y.y = mode*exp(sigma0*t)*(sin(t)*X.x+cos(t)*X.y);

     p0 = sigma0+k*(sigma0*sigma0+1)/gamma0;
     p1 = sigma1+(sigma1*sigma1+1)/(gamma1*k);
     X1.x = PHY[0][0]*(X.x-1.0)+PHY[0][1]*(X.y-p0)+1.0;
     X1.y = PHY[1][0]*(X.x-1.0)+PHY[1][1]*(X.y-p0)+p1;
     Y1.x = PHY[0][0]*(Y.x-1.0)+PHY[0][1]*(Y.y-p0)+1.0;
     Y1.y = PHY[1][0]*(Y.x-1.0)+PHY[1][1]*(Y.y-p0)+p1;
     r1 = sqrt(Y1.x*Y1.x+Y1.y*Y1.y);
     teta1 = atan(Y1.y/Y1.x);

     *Xu = sqrt(X1.x*X1.x+X1.y*X1.y)*exp(-sigma1*(pi+atan(X1.y)));

     if (Y1.y>=0.0)
          if (Y1.x<=0.0) *Yu = r1*exp(sigma1*(-teta1));
          else  *Yu = r1*exp(sigma1*(pi-teta1));
     else{
   if (Y1.x<=0.0){
               rX0 = X0*exp(sigma1*teta1);
               if (r1<=rX0) *Yu = r1*exp(sigma1*(2*pi-teta1));
               else{
                    printf("  Some points on the 'exit gate' hit L1 line before they
 are mapped\n");
                    printf("  onto the W1 plan by the poincare map P. The 1D-map's c
alculation\n");
                    printf("  then requires solving transcendental equation.\n");
                    printf("  Choose another dimensionless parameters for Chua's cir
cuit.\n\n");
                    exit(0);
      }
        }
          else{
               rX0 = X0*exp(sigma1*(pi-teta1));
               if (r1<=rX0) *Yu = r1*exp(sigma1*(pi-teta1));
               else{
                    printf("  Some points on the 'exit gate' hit L1 line before they
 are mapped\n");
                    printf("  onto the W1 plan by the poincare map P. The 1D-map's c
alculation\n");
                    printf("  then requires solving transcendental equation.\n");
                    printf("  Choose another dimensionless parameters for Chua's cir
cuit.\n\n");
                    exit(0);
      }
        }
          }
/**     if(*Yu<=X0){
          do{
          *Yu = *Yu * exp(sigma1*2.0*pi);
          }while(*Yu <= X0);
     }   **/
}

/** Calculation of u. The parameter mode indicates weither u=u+ or u=u-. **/

double IRT_f(t,mode)
double t;
int mode;
{
    double div;

     div = (exp(sigma0*t)*(cos(t)*B0.x-sin(t)*B0.y)+exp(gamma0*t)*(1.0-B0.x)-mode)
           /(exp(sigma0*t)*(cos(t)*(B0.x-1.0)-sin(t)*(B0.y-A0.y))+exp(gamma0*t)*(
.0-B0.x));
     return(div);
}


/** Print X array in fname file **/
void print_map(fname,X,nb)
char *fname;
double **X;
long nb;
{
     long i;
     FILE *fp;

     fp = fopen(fname,"w");
     for (i=0;i<=nb-1;i++) fprintf(fp,"%16.14f %16.14f\n",X[0][i],X[1][i]);
     fprintf(fp,"\n");
     fprintf(fp,"%16.14f %16.14f\n",0.,0.);
     fprintf(fp,"%16.14f %16.14f\n",X2max,X2max);
     fclose(fp);
}


#define EPS 2.0e-12
#define MAXM 100

/** Given the degree m and the m+1 complex coefficients a[0..m] of the polynomial *
/
/** a[0]+..+a[m](x..x), this routine successively calls laguer and finds all m    *
/
/** complex roots in roots[1..m]. The logical variable polish should be input as  *
/
/** TRUE(1).                                                                      *
/
void zroots(a,m,roots,polish)
fcomplex a[],roots[];
int m,polish;
{
 int jj,j,i;
 fcomplex x,b,c,ad[MAXM];
 void laguer();

 for (j=0;j<=m;j++) ad[j]=a[j];
 for (j=m;j>=1;j--) {
  x=Complex(0.0,0.0);
  laguer(ad,j,&x,EPS,0);
  if (fabs(x.i) <= (2.0*EPS*fabs(x.r))) x.i=0.0;
  roots[j]=x;
  b=ad[j];
  for (jj=j-1;jj>=0;jj--) {
   c=ad[jj];
   ad[jj]=b;
   b=Cadd(Cmul(x,b),c);
  }
 }
 if (polish)
```

```
  for (j=1;j<=m;j++)
   laguer(a,m,&roots[j],EPS,1);
 for (j=2;j<=m;j++) {
  x=roots[j];
  for (i=j-1;i>=1;i--) {
   if (roots[i].r <= x.r) break;
   roots[i+1]=roots[i];
  }
  roots[i+1]=x;
 }
}

#undef EPS
#undef MAXM



#define EPSS 2.e-14
#define MAXIT 100

/** Given the degree m and the m+1 complex coefficients a[0..m] of the polynomial **
/
/** a[0]+..+a[m](x..x), and given eps the desired fractionnel accuracy, and given **
/
/** a complex value x, this routine improves x by Laguerre's method until it con- **
/
/** verges to a root of the given polunomial. For normal use, polish should be    **
/
/** input as FALSE(0).                                                            **
/
void laguer(a,m,x,eps,polish)
fcomplex a[],*x;
int m,polish;
double eps;
{
 int j,iter;
 double err,dxold,cdx,abx;
 fcomplex sq,h,gp,gm,g2,g,b,d,dx,f,x1;
 void nrerror();

 dxold=Cabs(*x);
 for (iter=1;iter<=MAXIT;iter++) {
  b=a[m];
  err=Cabs(b);
  d=f=Complex(0.0,0.0);
  abx=Cabs(*x);
  for (j=m-1;j>=0;j--) {
   f=Cadd(Cmul(*x,f),d);
   d=Cadd(Cmul(*x,d),b);
   b=Cadd(Cmul(*x,b),a[j]);
   err=Cabs(b)+abx*err;
  }
  err *= EPSS;
  if (Cabs(b) <= err) return;
  g=Cdiv(d,b);
  g2=Cmul(g,g);
  h=Csub(g2,RCmul(2.0,Cdiv(f,b)));
  sq=Csqrt(RCmul((float) (m-1),Csub(RCmul((float) m,h),g2)));
  gp=Cadd(g,sq);
  gm=Csub(g,sq);
  if (Cabs(gp) < Cabs(gm))gp=gm;
  dx=Cdiv(Complex((float) m,0.0),gp);
  x1=Csub(*x,dx);
  if (x->r == x1.r && x->i == x1.i) return;
  *x=x1;
  cdx=Cabs(dx);
  if (iter > 6 && cdx >= dxold) return;
  dxold=cdx;
  if (!polish)
   if (cdx <= eps*Cabs(*x)) return;
 }
 nrerror("Too many iterations in routine LAGUER");
}

#undef EPSS
#undef MAXIT
```

Appendix E : The Amplitude Next Map of Chua's circuit.

In this appendix, we briefly study the dynamics of another one-dimensional map, the so-called Amplitude Next Map, uncorrelated with the map f presented in the previous sections. The Amplitude Next Map has been first constructed and applied to the analysis of the flow of a continuous-time dynamical system by Lorenz.

We make the observation that in Chua's circuit the maximum values of the time waveform components of the trajectory partly condition the geometric structure of the attractor : for example, at some critical values the trajectory accumulates on either a double scroll or a Rossler-type attractor.

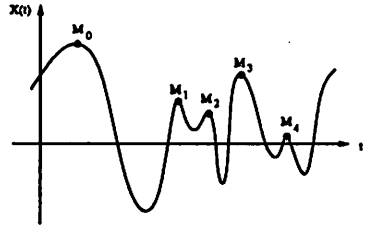


Figure 11: The construction of the Amplitude Next Map.

Denoting by M_n the n-th relative maximum of a given time waveform component of the trajectory (see Fig. 11), $x(t)$ for the purpose of this appendix, the Amplitude Next Map g is defined as follow :

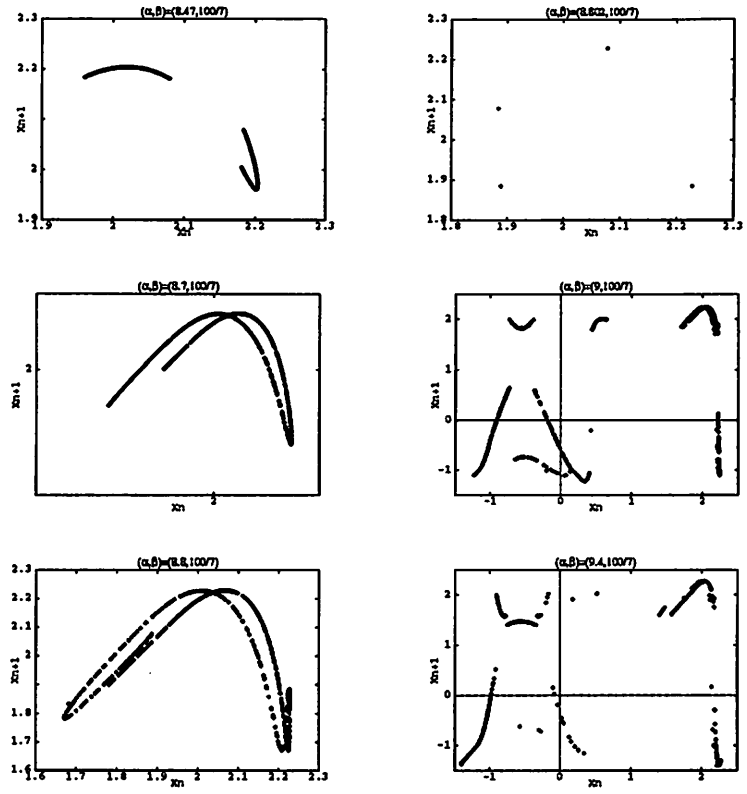
$$g(M_n) = M_{n+1} \tag{8}$$

Fig. 12 shows different patterns of the one-dimensional map g of Chua's circuit, obtained by fixing $(\beta, m_0, m_1) = (100/7, -8/7, -5/7)$ while varying α. Yet, in each non-periodic case, the map is clearly multivalued and defined on the union of distinct intervals. Therefore the derivation of results from the map g appears to be very uneasy and would require a far more in-depth analysis though it is not guaranteed that any valuable information on Chua's circuit would come out. For a stable periodic trajectory, the graph of the corresponding Amplitude Next Map is composed of a finite number of isolated points.

Eventually we provide here an algorithm for the construction of the Amplitude Next Map applied to Chua's circuit :

Step 1 Enter the time waveform component $x(t)$, $y(t)$ or $z(t)$ of the trajectory to be analysed, with a set of parameter values.

Step 2 Given arbitrary initial conditions in any region of the piecewise-linear vector field, initialize the linear system to be integrated.



(a) (b)

Figure 12: Some patterns of the Amplitude Next Map g : (a) corresponding to the Rossler type attractor and (b) to the Double Scroll attractor in Chua's circuit.

Step 3 Integrate the linear system.
If the trajectory crosses a boundary plane, find the crossing point and go to Step 2, with new initial conditions based on a boundary plane.

Step 4 While performing Step 2 and Step 3, localize the relative maximum values and store the data points (M_n, M_{n+1}).

Appendix F : C- Code routines for the construction of the Amplitude Next Map g.

```
/*****************************************************************************/
/**                                                                         **/
/**   This program is designed to study the one-dimensional Amplitude Next  **/
/**   Map, F, defined as follow :                                           **/
/**       Xn+1 = F(Xn)                                                      **/
/**   where Xn is the n-th mesured maximum of the time waveform of any      **/
/**   component x(t), y(t) or z(t) of the trajectory based at Xinit. Here   **/
/**   the dynamical system under study is Chua's circuit.                   **/
/**                                                                         **/
/**                        Version 1 - 07/20/1992                           **/
/**                          Author : Marc GENOT                            **/
/**                  Copyright : University of California                   **/
/*****************************************************************************/

#include <stdio.h>
#include <math.h>
#include "complex.c"
#include "util.c"

#define Dim 3
#define STEP 0.01

double alpha, beta, m0, m1;   /* circuit parameters */


/* argc=number of command-line arguments */
/* *argv[]=pointer to an array of character string */
main(int argc, char *argv[])
{
     double x (),y (),z ();
     static double (*dispatch[]) () = {x,y,z};
     double xinit[Dim];
     int option;
     void SEQUENCE();

     if(argc != 4) {
          printf("Usage: run <alpha> <beta> <option>\n");
          printf("       For <option> enter a component of the trajectory\n");
          printf("           (0 <-- x, 1 <-- y, 2 <-- z)\n");
         exit(0);
     }

     alpha=atof(argv[1]);
     beta=atof(argv[2]);
     m0 = -1.0/7.0;
     m1 = 2.0/7.0;
     xinit[0] = 0.1;
     xinit[1] = 0.1;
     xinit[2] = 0.1;
     option = atoi(argv[3]);

     SEQUENCE("reference",dispatch[option],xinit,option);
     printf("The Amplitude Next Map data are stored in file 'reference'.\n");
}



/** For an arbitrary initial condition, this procedure determine the sequence of reg
ions     **/
/** i visited by the trajectory, and the corresponding time intervals dti.
        **/
/** The initial region is return in 'initial_region' value and the successive time i
ntervals **/
/** in 'seq'
        **/
void SEQUENCE(fname,coordinate,xinit,option)
char *fname;
double (*coordinate)();
double *xinit;
int option;
{
     int region;                              /* region of current trajectory */
     double di,dj,t,tmax = 0.0;
     double xlast,xval,coor_val,coor_last = 0.0;
     double xn = 0.0;
     double a[Dim+1];
     fcomplex s[Dim];                         /* roots of cubic eq  */
     fcomplex k[Dim];                         /* coefficient of solution */
     FILE *fp;
     void initialization();
     double x(), y(), z(), find_crossing();

     fp = fopen(fname,"w");

     xlast = xinit[0];
     if(xinit[0] > 1.0) region = 2;
     else if(xinit[0] < -1.0) region = 0;
          else region = 1;
     initialization(region,xinit,a,k,s);
     do {
          t+=STEP;
          tmax+=STEP;
          xval = x(t,a,k,s);
          /** check to see if there is any crossing **/
          if((xval-1.0)*(xlast-1.0) < 0.0 || (xval+1.0)*(xlast+1.0) < 0.0){
               if(xval > 1.0) region = 2;
               else if(xval < -1.0) region = 0;
                    else region = 1;
               t = find_crossing(t,STEP,a,k,s);
               if(xval>0.0) xinit[0] = 1.0;
               else xinit[0] = -1.0;
               xinit[1] = y(t,a,k,s);
               xinit[2] = z(t,a,k,s);
               t = 0.0;
               xval = xinit[0];
               initialization(region,xinit,a,k,s);
               }
          xlast=xval;
     }while(tmax<1000.);


     tmax=0.0;
     do {
          t+=STEP;
          tmax+=STEP;
          xval = x(t,a,k,s);
          coor_val = coordinate(t,a,k,s);

          /** check to see if there is any crossing **/
          if((xval-1.0)*(xlast-1.0) < 0.0 || (xval+1.0)*(xlast+1.0) < 0.0){
               if(xval > 1.0) region = 2;
               else if(xval < -1.0) region = 0;
                    else region = 1;
               t = find_crossing(t,STEP,a,k,s);
               if(xval>0.0) xinit[0] = 1.0;
               else xinit[0] = -1.0;
               xinit[1] = y(t,a,k,s);
               xinit[2] = z(t,a,k,s);
```

```
            t = 0.0;
            xval = xinit[0];
            coor_val = xinit[option];
            initialization(region,xinit,a,k,s);
            }
        di = dj;
        dj = coor_val - coor_last;
        if ((di>0.0) && (di*dj<0.0)){
            if (xn != 0.0) fprintf(fp,"%12.10lf %12.10lf\n",xn,coor_last);
            xn=coor_last;
      }
        coor_last=coor_val;
        xlast=xval;
  }while(tmax<2000.0);
    fclose(fp);
}


void initialization(region,xinit,a,k,s)
int region;
double *xinit,*a;
fcomplex *k,*s;
{
    double z0[Dim];                /* reexpress initial conditions */
    void initabcd(), initial_condition(), findroot(), calc_coeff();


    initabcd(a,region);
    initial_condition(z0,a,xinit);
    findroot(a,s);
    calc_coeff(k,s,a,z0);
}



/** initialize the variable for 3rd order differential equation **/
/** for different region 0, 1, and 2 of the piecewise nonlinear **/
/** resistor                                                    **/
void initabcd(a,region)
double *a;
int region;
{
    a[1]= alpha;
    a[2] = -1.0*beta;

    /** first check the what is the current region **/
    if(region==0) {
        a[0] = -1.0*alpha*m1;
        a[3] = alpha*(m0-m1);
        }
    else if(region==1) {
        a[0] = -1.0*alpha*m0;
        a[3] = 0.0;
        }
    else if(region==2) {
        a[0] = -1.0*alpha*m1;
        a[3] = alpha*(m1-m0);
        }
    else printf("Error in initabcd()\n");
}



/** use bisection method to find crossing **/
double find_crossing(t,dt,a,k,s)
double t,dt,*a;
fcomplex *k,*s;
{
    double xval, xmid, tmid, rtb, deltat, xacc;

    /** specified root accuracy **/
    xacc = 10e-15;

    /** first determine what type of crossing **/
    xval = x(t,a,k,s);
    xmid = x(tmid = t-dt,a,k,s);

    if ((xval-1)*(xmid-1) < 0.0)  {              /** cross at x=1 **/
        xval -= 1.0;
        xmid -= 1.0;
        rtb = xval < 0.0 ? (deltat=tmid-t,t) : (deltat=t-tmid,tmid);
        do {
            xmid = x(tmid = rtb+(deltat*=0.5),a,k,s) - 1.0;
            if(xmid <= 0.0) rtb=tmid;
            if(fabs(deltat) < xacc || xmid == 0.0) return rtb;
            }while(1);
        }
    else {                                       /** cross at x=-1 **/
        xval += 1.0;
        xmid += 1.0;
        rtb = xval < 0.0 ? (deltat=tmid-t,t) : (deltat=t-tmid,tmid);
        do {
            xmid = x(tmid = rtb+(deltat*=0.5),a,k,s) + 1.0;
            if(xmid <= 0.0) rtb=tmid;
            if(fabs(deltat) < xacc || xmid == 0.0) return rtb;
            }while(1);
        }
}


/** calculate the initial conditions of Chua's circuit **/
/** with the appropriate coordinate basis              **/
void initial_condition(z0,a,xinit)
double *z0,*a,*xinit;
{
    /** calculate equivalent initial conditions **/
    z0[0] = xinit[2];
    z0[1] = a[2]*xinit[1];
    z0[2] = a[2]*(xinit[0] - xinit[1] + xinit[2]);
}


/** Use Laguerre's method to calculate the roots of the caracteristic equation **/
void findroot(a,s)
double *a;
fcomplex *s;
{
    int i;
    int polish = 1;
    fcomplex coeff[Dim+1];
    void zroots();

    for(i=0;i<=Dim;i++) coeff[i].i = 0.0;
    coeff[0].r = a[2]*a[0];
    coeff[1].r = -a[0]-a[1]-a[2];
    coeff[2].r = 1.0-a[0];
    coeff[3].r = 1.0;
    zroots(coeff,Dim,s,polish);
```

```
}

/** Calculate the coefficients, k[], explicitly **/
void calc_coeff(k,s,a,z0)
fcomplex *k,*s;
double *a,*z0;
{
     fcomplex M[3][3];             /* the Alternant Matrix */
     fcomplex c1, c2, c3, c4;      /* temporary storages */

     /* first check the type of roots */
     c1 = Cmul(Csub(s[0],s[1]),Csub(s[0],s[2]));
     M[0][0] = Cdiv(Cmul(s[1],s[2]),c1);
     c2.r = -1.0; c2.i = 0.0;
     M[0][1] = Cdiv(Cmul(c2,Cadd(s[1],s[2])),c1);
     c2.r = 1.0; c2.i = 0.0;
     M[0][2] = Cdiv(c2,c1);

     c1 = Cmul(Csub(s[1],s[2]),Csub(s[1],s[0]));
     M[1][0] = Cdiv(Cmul(s[0],s[2]),c1);
     c2.r = -1.0; c2.i = 0.0;
     M[1][1] = Cdiv(Cmul(c2,Cadd(s[0],s[2])),c1);
     c2.r = 1.0; c2.i = 0.0;
     M[1][2] = Cdiv(c2,c1);

     c1 = Cmul(Csub(s[2],s[0]),Csub(s[2],s[1]));
     M[2][0] = Cdiv(Cmul(s[0],s[1]),c1);
     c2.r = -1.0; c2.i = 0.0;
     M[2][1] = Cdiv(Cmul(c2,Cadd(s[0],s[1])),c1);
     c2.r = 1.0; c2.i = 0.0;
     M[2][2] = Cdiv(c2,c1);

     c1.r = z0[0]-a[3]/a[0]; c1.i = 0.0;
     c2.r = z0[1];           c2.i = 0.0;
     c3.r = z0[2];           c3.i = 0.0;
     k[0] = Cadd(Cadd(Cmul(c1,M[0][0]),Cmul(c2,M[0][1])),Cmul(c3,M[0][2]));
     k[1] = Cadd(Cadd(Cmul(c1,M[1][0]),Cmul(c2,M[1][1])),Cmul(c3,M[1][2]));
     k[2] = Cadd(Cadd(Cmul(c1,M[2][0]),Cmul(c2,M[2][1])),Cmul(c3,M[2][2]));
}


double x(t,a,k,s)
double t,*a;
fcomplex *k,*s;
{
     fcomplex x, temp1, temp2, temp3;
     fcomplex Cexp();

     temp1 = Cadd(RCmul(1/a[2],Cmul(s[0],s[0])),RCmul(1/a[2],s[0])); temp1.r -= 1.0;

     temp2 = Cadd(RCmul(1/a[2],Cmul(s[1],s[1])),RCmul(1/a[2],s[1])); temp2.r -= 1.0;

     temp3 = Cadd(RCmul(1/a[2],Cmul(s[2],s[2])),RCmul(1/a[2],s[2])); temp3.r -= 1.0;

     x = Cadd(Cadd(Cmul(temp1,Cmul(k[0],Cexp(RCmul(t,s[0])))),
               Cmul(temp2,Cmul(k[1],Cexp(RCmul(t,s[1]))))),
               Cmul(temp3,Cmul(k[2],Cexp(RCmul(t,s[2])))));
     x.r = x.r - a[3]/a[0];
     return(x.r);
}


double y(t,a,k,s)
double t,*a;
fcomplex *k,*s;
{
     fcomplex y;
     fcomplex Cexp();

     y = Cadd(Cadd(Cmul(s[0],Cmul(k[0],Cexp(RCmul(t,s[0])))),
               Cmul(s[1],Cmul(k[1],Cexp(RCmul(t,s[1]))))),
               Cmul(s[2],Cmul(k[2],Cexp(RCmul(t,s[2])))));
     y.r = y.r / a[2];
     y.i = y.i / a[2];
     return(y.r);
}


double z(t,a,k,s)
double t,*a;
fcomplex *k,*s;
{
     fcomplex z;
     fcomplex Cexp();

     z = Cadd(Cadd(Cmul(k[0],Cexp(RCmul(t,s[0]))),
               Cmul(k[1],Cexp(RCmul(t,s[1])))),
               Cmul(k[2],Cexp(RCmul(t,s[2]))));
     z.r = z.r + a[3]/a[0];
     return(z.r);
}


/** take the exponential of a complex number **/
fcomplex Cexp(x)
fcomplex x;
{
     double ereal;
     fcomplex result;

     ereal = exp(x.r);
     result.r = ereal * cos(x.i);
     result.i = ereal * sin(x.i);
     return (result);
}


#define EPS 2.0e-12
#define MAXM 100

/** Given the degree m and the m+1 complex coefficients a[0..m] of the polynomial *
/
/** a[0]+..+a[m](x..x), this routine successively calls laguer and finds all m    *
/
/** complex roots in roots[1..m]. The logical variable polish should be input as  *
/
/** TRUE(1).                                                                      *
/
void zroots(a,m,roots,polish)
fcomplex a[],roots[];
int m,polish;
{
 int jj,j,i;
 fcomplex x,b,c,ad[MAXM];
 void laguer();

 for (j=0;j<=m;j++) ad[j]=a[j];
 for (j=m;j>=1;j--) {
```

```
  x=Complex(0.0,0.0);
  laguer(ad,j,&x,EPS,0);
  if (fabs(x.i) <= (2.0*EPS*fabs(x.r))) x.i=0.0;
  roots[j-1]=x;
  b=ad[j];
  for (jj=j-1;jj>=0;jj--) {
   c=ad[jj];
   ad[jj]=b;
   b=Cadd(Cmul(x,b),c);
  }
 }
 if (polish)
  for (j=1;j<=m;j++)
   laguer(a,m,&roots[j-1],EPS,1);
 for (j=2;j<=m;j++) {
  x=roots[j-1];
  for (i=j-1;i>=1;i--) {
   if (roots[i-1].r <= x.r) break;
   roots[i]=roots[i-1];
  }
  roots[i]=x;
 }
}

#undef EPS
#undef MAXM


#define EPSS 2.e-14
#define MAXIT 100

/** Given the degree m and the m+1 complex coefficients a[0..m] of the polynomial **
/
/** a[0]+..+a[m](x..x), and given eps the desired fractionnel accuracy, and given **
/
/** a complex value x, this routine improves x by Laguerre's method until it con- **
/
/** verges to a root of the given polunomial. For normal use, polish should be    **
/
/** input as FALSE(0).                                                            **
/
void laguer(a,m,x,eps,polish)
fcomplex a[],*x;
int m,polish;
double eps;
{
 int j,iter;
 double err,dxold,cdx,abx;
 fcomplex sq,h,gp,gm,g2,g,b,d,dx,f,x1;
 void nrerror();

 dxold=Cabs(*x);
 for (iter=1;iter<=MAXIT;iter++) {
  b=a[m];
  err=Cabs(b);
  d=f=Complex(0.0,0.0);
  abx=Cabs(*x);
  for (j=m-1;j>=0;j--) {
   f=Cadd(Cmul(*x,f),d);
   d=Cadd(Cmul(*x,d),b);
   b=Cadd(Cmul(*x,b),a[j]);
   err=Cabs(b)+abx*err;
  }
  err *= EPSS;
  if (Cabs(b) <= err) return;
  g=Cdiv(d,b);
        g2=Cmul(g,g);
  h=Csub(g2,RCmul(2.0,Cdiv(f,b)));
  sq=Csqrt(RCmul((float) (m-1),Csub(RCmul((float) m,h),g2)));
  gp=Cadd(g,sq);
  gm=Csub(g,sq);
  if (Cabs(gp) < Cabs(gm))gp=gm;
  dx=Cdiv(Complex((float) m,0.0),gp);
  x1=Csub(*x,dx);
  if (x->r == x1.r && x->i == x1.i) return;
  *x=x1;
  cdx=Cabs(dx);
  if (iter > 6 && cdx >= dxold) return;
  dxold=cdx;
  if (!polish)
   if (cdx <= eps*Cabs(*x)) return;
 }
 nrerror("Too many iterations in routine LAGUER");
}

#undef EPSS
#undef MAXIT
```

References

[1] T. Matsumoto. A chaotic attractor from chua's circuit. *IEEE Transactions on Circuits and Systems, CAS-31*, 12:1055–1048, 1984.

[2] L. O. Chua. The genesis of chua's circuit. *Archiv fur Elektronik und Ubertragung stechnik*, 46(4):187–257, 1992.

[3] M. P. Kennedy. Robust op amp realization of chua's circuit. *Frequenz*, 46:66–80, 1992.

[4] M. Komuro L. O. Chua and T. Matsumoto. The double scroll family. *IEEE Transactions on Circuits and Systems, CAS-33*, 33:1072–1118, 1986.

[5] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems.* Addison-Wesley, 1989.

[6] T. Y. Li and J. A. Yorke. Period three implies chaos. *Amer. Math. Montly*, 82:985–992, 1975.

[7] J. P. Eckmann P. Collet. *Iterated Maps on the Interval as Dynamical Systems.* Progress in Physics, 1980.

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

[9] L. O. Chua and I. Tickonicky. 1-d map for the double scroll family. *IEEE Transactions on Circuits and Systems, CAS-31*, 38:233–243, 1991.

[10] D. P. George. Bifurcations in a piecewise-linear system. *Physical Letters A*, 112:17–21, 1986.

[11] P. Glendinning and C. T. Sparrow. Local and global behavior near homoclinic orbits. *Journal of Statistical Physics*, 35:645–696, 1984.

[12] R. Tokunaga M. Komuro and A. Hotta T. Matsumoto, L. O. Chua. Global bifurcation analysis of the double scroll circuit. *International Journal of Bifurcation and Chaos*, 1:139–182, 1991.