Institut National des Sciences Appliquées
of Toulouse

The University of Tokushima

Numerical Method of Bifurcation Analysis for Hybrid Systems

by

Quentin BRANDON

Supervisors:

Danièle Fournier-Prunaret

Tetsushi Ueta

September 2009

Abstract

In the field of dynamical system analysis, piecewise-smooth models have grown in popularity due to there greater flexibility and accuracy in representing some hybrid systems in applications such as electronics or mechanics. Hybrid dynamical systems have two sets of variables, one which evolve in a continuous space, and the other in a discrete one.

Most analytical methods require the orbit to be smooth during objective intervals, so that some special treatments are inevitable to study the existence and stability of solutions in hybrid dynamical systems.

Based on a piecewise-smooth model, where the orbit of the system is broken down into locally smooth pieces, and a hybrid bifurcation analysis method, using a Poincare map with sections ruled by the switching conditions of the system, we review the analysis process in details.

Then we apply it to various extensions of the Alpazur oscillator, originally a non-smooth 2-dimension switching oscillator. The original Alpazur oscillator, as a simple nonlinear switching system, was a perfect candidate to prove the efficiency of the approach. Each of its extensions shows a new scenario and how it can be handled, in order to illustrate the generality of the model. Finally, and in order to show more of the implementation we used for our own computer-based analysis tool, some of the most relevant numerical methods we used are introduced.

It is noteworthy that the emphasis has been put on autonomous systems because the treatment of non-autonomous ones only requires a simplification (no time variation). This study brings a strong and general framework for the bifurcation analysis of nonlinear hybrid dynamical systems, illustrated by some results. Among them, some interesting local and global properties of the Alpazur Oscillator are revealed, such as the presence of a cascade of cusps in the bifurcation diagram. Our work resulted in the implementation of an analysis tool, implemented in `C++`, using the numerical methods that we chose for this particular purpose, such as the numerical approximation of the second derivative elements in the Jacobian matrix.

Acknowledgment

This thesis is the final achievement of 4 years of studies in Japan. The opportunity to engage in this dual PhD. program actually comes with both a great academic, and human experience. So I would like to thank all the people who helped me and made this adventure possible.

First, I want to thank my supervisors, Tetsushi Ueta and Danièle Fournier-Prunaret, for believing in me and supporting me from beginning to the end.

I am also very grateful to the reviewers of this thesis, Jean-Pierre Barbot and Hiroyuki Kitajima, who took the time to read my work, and to give me extremely helpful feedback. This also applies to the jury of my defense who kindly listened and asked very constructive questions.

Of course, all this could not have been possible without both the University of Tokushima and INSA Toulouse. I thank all the teachers, and all the students. Also I am particularly obliged to the Japanese ministry of education, culture, sports, science and technology (MEXT), which financed my studies in Japan through their scholarship program.

I thank my friends, most of whom I met here in Tokushima, for all the great times we had and the help so many of them provided me. This group includes the international students, my lab-mates and many others who will recognize themselves.

Last but not the least, I thank my family, for supporting me and visiting me up to the most remote places on this planet. A special mention to my wife as she is the most precious treasure I got to find in Tokushima, where life became twice as beautiful since the day our paths came together.

Contents

Chapter 1

Introduction

The analysis of dynamical systems, particularly complex systems, dramatically evolved at the introduction of the first computer-based mathematical tools, only a couple decades ago. Meanwhile, the discovery of seemingly random behavior in dynamical systems lead to the further study of what became the chaos theory. Such chaotic systems display a variety of behaviors: when described as oscillators, the same system may have stable orbits, but changes applied to one or several parameters could induce changes of these orbits to states of equilibrium, change in the number of periods, or even reach instability. Those changes are called bifurcations, and when the dynamics of the system become so sensitive to the initial conditions that the evolution of the system can no longer be predicted on the long run, while remaining in a finite interval, it belongs to the group of chaotic systems. The next question that comes naturally when trying to understand chaotic systems is what are the causes of chaotic motions. As it appears that bifurcations occurring in the parameters space trigger the transition from and to chaotic dynamics, they became of major interest. While bifurcation types are determined by their effect on the system, they can also be grouped by how they occur, which is usually directly related to their causes. Without getting too much into details, the two groups usually considered are local bifurcations, where the topological change in the phase portrait is limited to a small neighborhood of the bifurcation fixed point; and global bifurcations where such variation cannot be delimited.

In order to study these phenomena among dynamical systems, most existing mathematical approaches have been established on the base of simple models either in a continuous or discrete time space. However, in many cases of "real-world problems" of dynamical systems analysis, purely continuous or discrete models are insufficiently precise approximations. Systems showing continuous and discrete dimensions exist in many domains such as mechanical (apparition of frictions, impacts) or electronic systems (switching component). Though the solution function of such systems is sometimes continuous, it presents points of non-derivability where discrete changes occur, thus falling into the category of piecewise smooth functions. Some argue that there is no such thing as discrete changes in natural dynamics. This affirmation is sup-

ported by the idea that any event appears smooth if one can measure its appearance at the appropriate time scale (an impact observed in slow motion can appear smooth if slowed-down enough). When modeling a system on the other hand, it is important to consider the overall timescale of its dynamics, and when an event occurs fast enough, it becomes relevant to consider it discrete.

Some of the most widely studied applications that use such models are usually related to electrical engineering, such as power converters investigated as piecewise smooth systems by di Bernardo & Tse [1], Tse [2], Banerjee & Chakrabarty [3] or Banerjee & Karthick *et al* [4]; or some PLL models as introduced by Acco [5], referring to this type of model as "hybrid sequential."As for mechanical systems, a general methodology was introduced by Wiercigroch & De Kraker [6]. Also, complete and up-to-date information about piecewise-smooth dynamical systems in general can be found in the book by di Bernardo *et al* [7].

On the other hand, little work has been done in the bifurcation analysis of nonlinear piecewise-smooth models, except by Kawakami & Lozi [8] and Kousaka *et al* [9], who later introduced some chaos control applications [10]. As for linear piecewise-smooth models, they can be analyzed by using rigorous analytical methods, so exact solutions are obtained and used for analysis of the total dynamical behavior, as did Kabe *et al* [11]. However the nonlinear characteristics of many systems make this option possible for approximations only. At this point, computer tools become of great help. Despite the inherent limitation in precision and the need to integrate continuous variations using discrete methods, the performance and results are mostly satisfactory for the purpose of simulation and analysis, and often better than the analytical alternative, at least with regard to performance.

Dankowicz [12] proposes coarsening smooth vector fields into piecewise-smooth approximations which seems to be a good adaptation of traditional methods to systems subject to grazing bifurcations. Combined with our analysis method, we can imagine applications to a much wider range of dynamical systems. Hiskens and Pai [13] rely on the Newton method to obtain a discrete approximation and study the system's sensitivity at switching points in the scope of trajectory sensitivity analysis which can give a good insight of the switching aspect of such systems, but need to be completed by other methods to obtain a general bifurcation analysis, particularly for parameters which do not influence the switching conditions.

On our side, we focus on numerical methods, with the aim of detailing how to make a computer based tool for such analysis. The hardware used was a multi-core workstation, so we tried to keep the possibility of optimization in mind, particularly since the last few years propelled parallel computing to the front of the scene as the present and future of computers. Even so, there was no supercomputer involved and any mainstream computer is able to sustain a proper performance level for the program derived from this analysis framework to deliver precise results within a more than acceptable time lapse.

On the mathematical aspect, differential equations are used to model nonlinear

autonomous systems for each of their discrete states. We paid a particular attention to systems where the variable domains determine the state of the system, which in turn determines the differential equations that characterize the system locally. This is a very flexible scenario which can be adapted to other cases, such as time-based switching where all is required is a simplification.

By fixing the state sequence, we fix the boundaries of the problem. Therefore this type of model belongs to the "hybrid sequential" category.

We propose using a Poincaré map in order to obtain a discrete map and conduct the bifurcation analysis using conventional methods: in other words, a hybrid method to solve a hybrid problem. This complements and update the work of Kousaka and al. [9] which introduced the originally proposed method for piecewise smooth system analysis. We first review the method, detailing each step of the process from modeling, to solving fixed and bifurcation points, using a piecewise analysis and a discrete map to combine the local results. Next we apply it to multiple versions of the Alpazur oscillator in order to illustrate concrete analysis and results. Finally, we consider some key algorithms we used as part of the computer based tool we implemented. The major improvement we introduce concerns a numerical alternative approach to obtain the Jacobian matrix in order to avoid the complexity of the analytical method suggested in [9], which clearly presented that point as a major candidate for improvement. We also consider non trivial Poincare section with variable switching conditions illustrated by affine, non-linear, and even non-smooth switching thresholds in the Alpazur oscillator. Regarding the results, the analysis of the Alpazur oscillator reveals an unusual bifurcation structure: the interactions between the equilibrium point at some state and the corresponding switching condition generate a fractal bifurcation structure, with an infinite number of bifurcation curves focusing towards a limit set. We consider the line, constituting this limit set, as a global bifurcation line. It appears to involve each variant of the Alpazur oscillator we have analyzed so far. Similar structures have been found and studied by Carcasses & Mira [14] and Mira & Taha [15], in piecewise linear dynamical systems, putting the accent on the existence of "cascades of cusps".

Chapter 2

Bifurcation analysis method for piecewise-smooth dynamical systems

Let us first make a few remarks about the context and state of mind in which our work has been carried out. Our intend is to present a bifurcation analysis approach that overcomes the shortcomings of methods purely discrete or purely continuous oriented. It should also be possible to implement this method in a computer program. This means we need a description model as flexible as possible, in order to handle as many kinds of systems as possible, while limiting the model complexity to its minimum for technical and usability reasons.

As we will see later on, the key steps of our approach are based on a Poincaré map, and Newton's method. We will almost exclusively consider local bifurcations for two reasons: eigenvalues on which most of our study is based, are only relevant to local bifurcations; the large variety of global bifurcations and their very tight dependency to the system itself require case by case studies, impractical when conceiving a general-purpose algorithm.

In order to make the analysis possibilities flexible, yet technically achievable in a reasonable amount of time and effort, we chose to not only model the system, but also part of its local behavior. Such boundaries make the analysis simpler and more targeted, and though it requires some minimum knowledge about the system prior to processing its model, it helps speeding and focus the analysis to some of its specific properties.

From now on, by model, we will mean the qualitative description of the system and the necessary information about its local behavior.

2.1 Differential equations model featuring switching thresholds

Let us consider a system described by a set of differential equations such that for each state i:

$$\frac{dX}{dt} = f_i(X), \quad i = 1, \cdots m, \quad \text{where} \quad X(t) = (x_1, \cdots x_n) \in R^n. \tag{2.1}$$

Each state dependent function f_i is piecewise-smooth. Within each state i, there is a local solution function that we have to obtain through numerical integration, and which can be written as:

$$X(t) = \varphi_i(t, X_{i-1}) \quad \text{with} \quad X(0) = X_{i-1}, \tag{2.2}$$

where X_{i-1} is the initial value at state i.

In order to complete our model, we will need to fix a sequence of states that will describe a period of our system, and the switching condition from each state to the next one. Here, we will focus on the case of autonomous systems. Now, a couple of remarks can be made: first, this can easily be adapted to non-autonomous (or by extension autonomously-hybrid) systems given that the case of a switch determined by time is a much simpler and straight forward case as we will see further; second, though we do not mention it in the equations, note that applying some linear transformation is possible when switching from one state to another, which is often the case in concrete systems.

We can now consider a Poincaré map, placing its sections at the switching points X_i. In the case of an autonomous system, the switching condition can be expressed as a function of the system variables. We write the function determining the switching condition from state i to the next one: $q_i(X) = 0$.

According to Kawakami and Lozi [8], the sequence of transformation that will compose the Poincaré map is therefore expressed as

$$\begin{aligned} &\Pi_i = \{X_i \in R^n \mid q_i = 0\} \\ &T_i : \ \Pi_{i-1} \to \Pi_i \\ &\qquad X_{i-1} \mapsto X_i = \varphi_i(\tau_i, X_{i-1}). \end{aligned} \tag{2.3}$$

Using such definition, we can perform a local analysis over each segment of orbit delimited by those Poincaré sections. Then, the Poincaré mapping is defined as a differentiable map:

$$\begin{aligned} &T = T_m \circ \cdots \circ T_1 \circ T_0. \\ &T : \ \Pi_0 \to \Pi_0 \\ &\qquad X_0 \mapsto X_m = \varphi(\tau, X_0). \end{aligned} \tag{2.4}$$

The last step is to use a projection p based on q_m in order to handle a discrete approximation of the periodic orbit:

$$\begin{array}{rcl} p & : & \Pi_0 \to \Sigma_0 \\ & & X \mapsto U \end{array} \qquad \text{hence} \quad T_l = p \circ T \circ p^{-1}. \tag{2.5}$$

Figure 2.1 gives an abstract representation of such model.

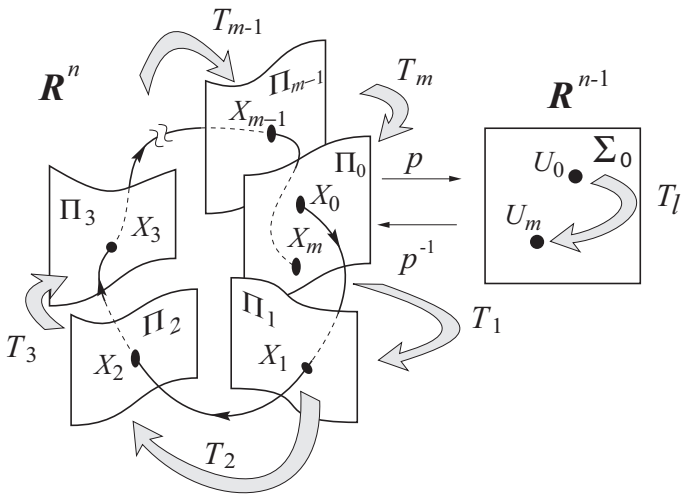


Figure 2.1: Abstract representation of the Poincaré map

Note that one simple way of handling k-periodic orbits is to modify this model to have a state sequence repeating its original pattern k times. While being a simple approach it is also very scalable and easy to automatize.

2.2 Integration of the solutions

Whether we want to run a simulation of the system or study its local properties, we will need to use a numerical integration method. For precision and performance reasons, we generally use a Runge-Kutta based method which we will detail later on in the section 4.2. For now, we will focus on the mathematical aspect more then the numerical one. With respect to the previously exposed model, we simply integrate X using (2.1).

If the system model, parameters and initial conditions are right, we are able to integrate each portion of orbit within each state; the last point of each piece of orbit becoming the initial point of the next state after the switch occurs. Depending on the system, the switching operation may alter the solution's continuity, and most of the time make it at least non-smooth, but this affects only the Poincaré map. Each piece of the orbit corresponding to a single state is smooth and derivable. This way, we obtain $X_m = \varphi(\tau, X_0)$.

In order to conduct further analysis, as we will see later on, we also need the derivatives of this solution. We will use two different approaches, the first one being based on an analytically obtained expression of the Jacobian of f_i, and another one more numerical which will be detailed in the section 4.1.

We proceed by integrating the partial derivatives of the differential equations. In the case of $\partial\varphi_i(\tau_i, X_{i-1})/\partial X_{i-1}$:

$$\frac{d}{dt}\frac{\partial\varphi_i}{\partial X_{i-1}} = \frac{\partial f_i}{\partial X}\frac{\partial\varphi_i}{\partial X_{i-1}} \quad \text{with} \quad \left.\frac{\partial\varphi_i}{\partial X_{i-1}}\right|_{t=0} = I_n, \tag{2.6}$$

where I_n is the identity matrix.

Note also that for the sake of readability we replace $\varphi_i(\tau_i, X_{i-1})$ by φ_i. When necessary, this will be pointed out in the following sections.

When taking into account the influence of a particular parameter λ of f_i, the idea is the same:

$$\frac{d}{dt}\frac{\partial\varphi_i}{\partial\lambda} = \frac{\partial f_i}{\partial X}\frac{\partial\varphi_i}{\partial\lambda} + \frac{\partial f_i}{\partial\lambda} \quad \text{with} \quad \left.\frac{\partial\varphi_i}{\partial\lambda}\right|_{t=0} = 0 \quad \text{zero vector of } n \text{ elements.} \tag{2.7}$$

We use those elements as an approximation of the first derivative of the solution, assuming that its local behavior is nearly linear.

The next section explains how.

2.3 Poincaré map and Newton method for fixed points

Using a phase portrait plotter one can appreciate the system's behavior and find a set of initial conditions and parameter values worth considering further.

The next step is to find a closed orbit, that is, from the perspective of the Poincaré map, a fixed point. Using the Poincaré map expressed earlier, we will look for a fixed point within Σ_0.

$$U_m = T_l(U_0) = U_0. \tag{2.8}$$

We then use the Newton method to refine the given set of initial conditions and parameter values. This assumes the input is close enough to the solution to fall within the convergence domain.

Let us consider the case of corrections applied to U_0, we then need to apply the Newton method using $\partial U_m/\partial U_0$. It can be derived from $\partial X_m/\partial X_0$ thanks to p.

$$\frac{\partial X_m}{\partial X_0} = \prod_{i=1}^{m} \frac{\partial X_i}{\partial X_{i-1}}. \tag{2.9}$$

Now we will use the local Jacobian of the solution, integrated as in section 2.2, to compute each $\partial X_i/\partial X_{i-1}$:

$$\frac{\partial X_i}{\partial X_{i-1}} = \frac{\partial \varphi_i}{\partial X_{i-1}} + \frac{dX_i}{dt}\frac{\partial \tau_i}{\partial X_{i-1}}. \tag{2.10}$$

The expression of $\partial \tau_i/\partial X_{i-1}$ depends on the switching condition. It is important to stress two things:

- This element takes the variation of τ a unit of time, which shows how much easier things get when dealing with non-autonomous systems.
- For systems with switch induced phenomena (such as jumps in some neuron models like the one introduced by Izhikevich [16]), this is the equation that requires adjustments.

In most of the systems we have been studying, the expression of $\partial \tau_i/\partial X_{i-1}$ takes into account the following terms and functions: $\partial \varphi_i/\partial X_{i-1}$, $q_i(X)$, and $f_i(X)$.

The corrected value of U_0 is given by:

$$U_0^* = U_0 - \left(\frac{\partial U_m}{\partial U_0} - I_{n-1} \right)^{-1} (U_m - U_0). \tag{2.11}$$

When attempting to apply a correction to one of the system parameters, let us call it λ, to find fixed points, the process is the same. One of the elements of U_0 has to become static, and the Jacobian matrix completed with the partial derivative $\partial U_m/\partial \lambda$.

Because it can be hard to explore a system's behavior by randomly changing initial values and parameters and plot the result, we usually find a single fixed point, then find all the contiguous ones by applying small variations to a chosen parameter. We obtain a curve of fixed points in the U_0/λ space, which usually extends until reaching a collision bifurcation or some other global bifurcation.

As we will see in section 2.4, we will pay a particular attention to the eigenvalues, which can be monitored along such fixed points curve in order to find some candidate bifurcation points to be refined using the following approach.

2.4 Critical parameter values at bifurcation points

The study of local bifurcations can be performed numerically. We can determine what eigenvalue μ to seek based on the decided bifurcation type. Then we need to find precise values of U_0 ans λ to have:

$$\chi_l(\mu) = \det\left(\frac{\partial U_m}{\partial U_0} - \mu I_{n-1}\right) = 0, \tag{2.12}$$

where χ_l is the characteristic equation.

This added to the fixed point constraint, and we obtain the following expression:

$$F(U_0, \lambda) = \begin{bmatrix} U_m - U_0 \\ \chi_l(\mu) \end{bmatrix} = 0. \tag{2.13}$$

In order to compute the appropriate corrections of U_0 and λ, we will need the following type of Jacobian matrix:

$$\begin{bmatrix} \dfrac{\partial U_m}{\partial U_0} - I_{n-1} & \dfrac{\partial U_m}{\partial \lambda} \\ \dfrac{\partial \chi_l}{\partial U_0} & \dfrac{\partial \chi_l}{\partial \lambda} \end{bmatrix} \tag{2.14}$$

In order to obtain a bifurcation diagram, we actually choose two parameters λ_1 and λ_2, then trace a bifurcation line by applying small variations to one parameter while correcting the other one along with U_0.

To reduce the number of iterations necessary to convergence at each computation, prediction algorithms can be integrated to the tracing process as we will see in section 4.3.

Also, in order to handle the various shapes of bifurcation curves possible, we sometimes need to switch role between λ_1 and λ_2, or even correct both at a time while fixing one element of U_0, like in the case of cusp points. The Jacobian matrix then needs to be adapted accordingly.

You will notice that Eq. (2.14) requires the derivation of the characteristic equation χ_l, which is a function of $\partial U_m/\partial U_0$ (sometimes also noted DT_l for readability). This means we need to compute the second derivative of the solution in order to obtain $\partial^2 U_m/\partial U_o^2$ or $\partial^2 U_m/\partial U_o \partial \lambda$.

While deriving (2.10) and integrating each of the necessary elements has been proved possible by Kousaka [10], it also appeared to be a painful process requiring many, indeed elegant, yet time consuming tasks on both the analytical and numerical sides. Because we do not mind the computer working extra hard to relieve the user of a few pages of manual derivation, we developed a numerical approach that does exactly that and introduce it in section 4.1.

Chapter 3

Alpazur oscillator

The Alpazur oscillator is a power electronic circuit introduced by Kawakami and Lozi [8]. It is composed of a Rayleigh oscillator unit and a dc power supply controlled by a switch (itself controlled through a feedback loop). Though a nonlinear resistor is rare in real-world applications, this simple circuit is:

- nonlinear
- piecewise-smooth
- chaotic

It can be both easily modeled and easily implemented making it a perfect academic case for the study of autonomous piecewise-smooth nonlinear dynamical systems.

In this chapter, we will review the original Alpazur oscillator, introduce and study some extended systems based on it, and analyze the results we obtained.

3.1 Review of the original Alpazur oscillator

Alpazur oscillator was introduced in 1992 by Kawakami and Lozi in [8]. Its bifurcations have been analyzed using numerical methods by Kousaka in [10], who confirmed his results with an actual circuit implementation.

3.1.1 Presentation

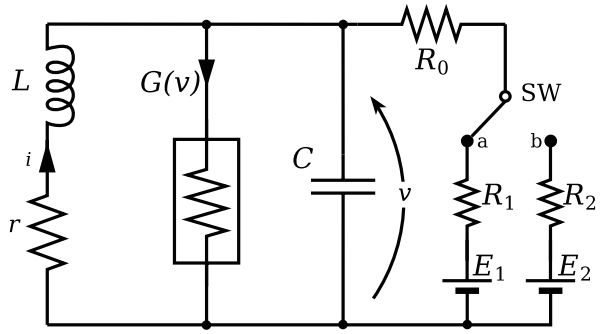


Figure 3.1: Electronic implementation of the Alpazur oscillator.

A simple RLC oscillator is connected to a nonlinear resistor G and a power supply. The characteristics of the latter change discretely based on the position of the switch SW (as shown in Fig. 3.1.)

The variables considered are i and v, the current running through the coil and the voltage at the capacitor C. In the original Alpazur, the switching rule of SW is a set of thresholds of v. This implies a discrete state, hence the hybrid caracteristic of this system: two continuous dimension and a discrete one.

$$\begin{cases} L\dfrac{di}{dt} &= -ri - v \\ C\dfrac{dv}{dt} &= i - g(v) + \dfrac{E_j - v}{R_0 + R_j} \end{cases} \qquad \text{with} \quad j = 1, 2. \tag{3.1}$$

The characteristic of the nonlinear-resistor is chosen as:

$$g(v) = -a_1 v + a_3 v^3, \quad \text{where} \quad a_1, a_3 > 0 \tag{3.2}$$

In order to model the system, the following variable change and assumptions will be made:

$$X = \begin{pmatrix} x \\ y \end{pmatrix} \quad \begin{array}{llll} x & = i\sqrt{L} & y & = v\sqrt{C} \\ \tau & = \dfrac{t}{\sqrt{LC}} & r & = r\sqrt{\dfrac{C}{L}} \\ b & = a_1\sqrt{\dfrac{L}{C}} & c & = \dfrac{3a_3}{C}\sqrt{\dfrac{L}{C}} \\ A_j & = \sqrt{\dfrac{L}{C}}\dfrac{1}{R_0+R_j} & B_j & = \sqrt{L}\dfrac{E_j}{R_0+R_j} \end{array} \tag{3.3}$$

Finally, we pick the following parameter values $b = c = 1$.

This results in the equation set:

$$\begin{cases} \dfrac{dx}{d\tau} & = -rx - y \\ \dfrac{dy}{d\tau} & = x + (1 - A_i)y - \dfrac{1}{3}y^3 + B_i \end{cases} \quad \text{with the discrete state} \quad i = 1, 2. \tag{3.4}$$

The switching conditions are:

$$q_i(X) = y - h_i, \qquad \text{with} \qquad h_1 = -1, \quad \text{and} \quad h_2 = -0.1. \tag{3.5}$$

Consequently, this system presents an hysteresis (see Fig. 3.2,) potentially yielding stable chaotic orbits (as shown in Fig. 3.3.)

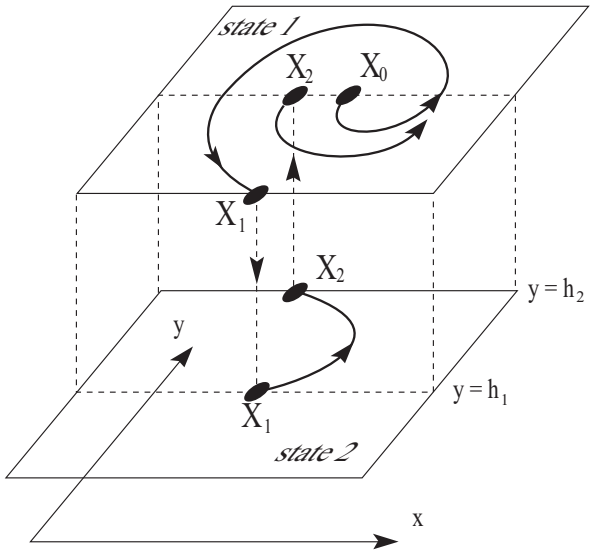


Figure 3.2: Typical period of the Alpazur oscillator in the hybrid space, refer to (3.6) for the switching point notation X_i.

In some parameter regions, this system exhibits chaotic orbits. By tuning parameters A_i and B_i through arbitrary values, we obtain chaotic behavior as seen in Fig.3.3.
$r = 0.1 \quad A_1 = 0.2 \quad A_2 = 2.0 \quad B_1 = -0.2 \quad B_2 = 1.0$

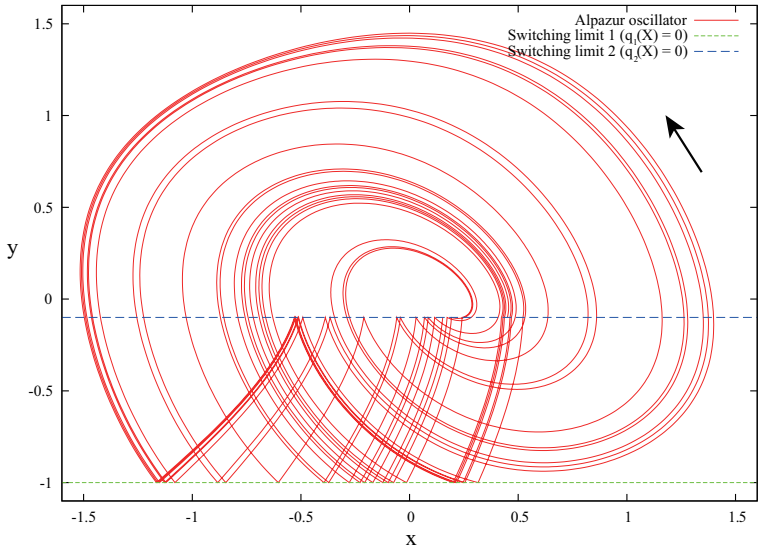


Figure 3.3: Phase portrait of a chaotic orbit. The chaotic behavior is due to the presence of a discrete state on top of the two continuous dimensions.

3.1.2 Analysis formulation

We naturally consider the Poincaré map as follows:

$$\begin{array}{rrcl} T_1: & \Pi_0 & \to & \Pi_1 \\ & X_0 & \mapsto & X_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \varphi_1 \\ \phi_1 \end{pmatrix} (\tau_1, X_0) \\ T_2: & \Pi_1 & \to & \Pi_0 \\ & X_1 & \mapsto & X_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \varphi_2 \\ \phi_2 \end{pmatrix} (\tau_2, X_1) \\ T = T_2 \circ T_1: & \Pi_0 & \to & \Pi_0 \\ & X_0 & \mapsto & X_2 \end{array} \tag{3.6}$$

where $\Pi_0 = \{X \in \Re^2 | q_2(X) = 0\}$ and $\Pi_1 = \{X \in \Re^2 | q_1(X) = 0\}$.

Considering the switching conditions, we define the projection p:

$$\begin{array}{rrcl} p: & \Pi_0 & \to & \Sigma_0 \\ & X & \mapsto & x \end{array} \tag{3.7}$$

The equation set can be written as follows for the state i:

$$\left\{ \begin{array}{lll} \dfrac{dx}{d\tau} & = f_i(x,y) & = -rx - y \\ \dfrac{dy}{d\tau} & = g_i(x,y) & = x + (1 - A_i)y - \dfrac{1}{3}y^3 + B_i \end{array} \right. \quad \text{with} \quad i = 1, 2. \tag{3.8}$$

3.1.3 Fixed points

The first step to conduct our analysis is determining fixed points.

As previously explained, the problem of fixed points is finding x_0 so that:

$$x_2 - x_0 = 0. \tag{3.9}$$

In order to achieve the appropriate correction using Newton's method, we need to compute:

$$\frac{\partial x_2}{\partial x_0} = \frac{\partial x_2}{\partial x_1}\frac{\partial x_1}{\partial x_0}. \tag{3.10}$$

For each State i we compute:

$$\begin{aligned} \frac{dx_i}{dx_{i-1}} &= \frac{\partial \varphi_i}{\partial x_{i-1}} + f_i(x_i, y_i)\frac{\partial \tau_i}{\partial x_{i-1}} \\ \text{where} \qquad \frac{\partial \tau_i}{\partial x_{i-1}} &= \frac{-\dfrac{\partial \phi_i}{\partial x_{i-1}}}{g_i(x_i, y_i)}, \end{aligned} \tag{3.11}$$

numerically integrate the required elements:

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x_i \\ y_i \end{bmatrix} &= \begin{bmatrix} f_i(x,y) \\ g_i(x,y) \end{bmatrix} \quad \begin{aligned} &\text{State 1: from } \begin{bmatrix} x_0 \\ b \end{bmatrix} \\ &\text{State 2: from } \begin{bmatrix} x_1 \\ h \end{bmatrix} \end{aligned} \\ \frac{d}{dt}\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix} &= \begin{bmatrix} \dfrac{\partial f_i}{\partial x} & \dfrac{\partial f_i}{\partial y} \\ \dfrac{\partial g_i}{\partial x} & \dfrac{\partial g_i}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix} \\ \text{where } \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix}_{t=0} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \end{aligned} \tag{3.12}$$

Note that we could drop $\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial \phi_i}{\partial y_{i-1}} \end{bmatrix}$ since y_i values are fixed by the switching limits (3.5). We now use the Newton method to compute x_0', the correction to be applied:

$$x_0' = x_0 - \frac{x_2 - x_0}{\dfrac{\partial x_2}{\partial x_0} - 1}. \tag{3.13}$$

We obtain a diagram of fixed points (such as in Fig. 3.4.)

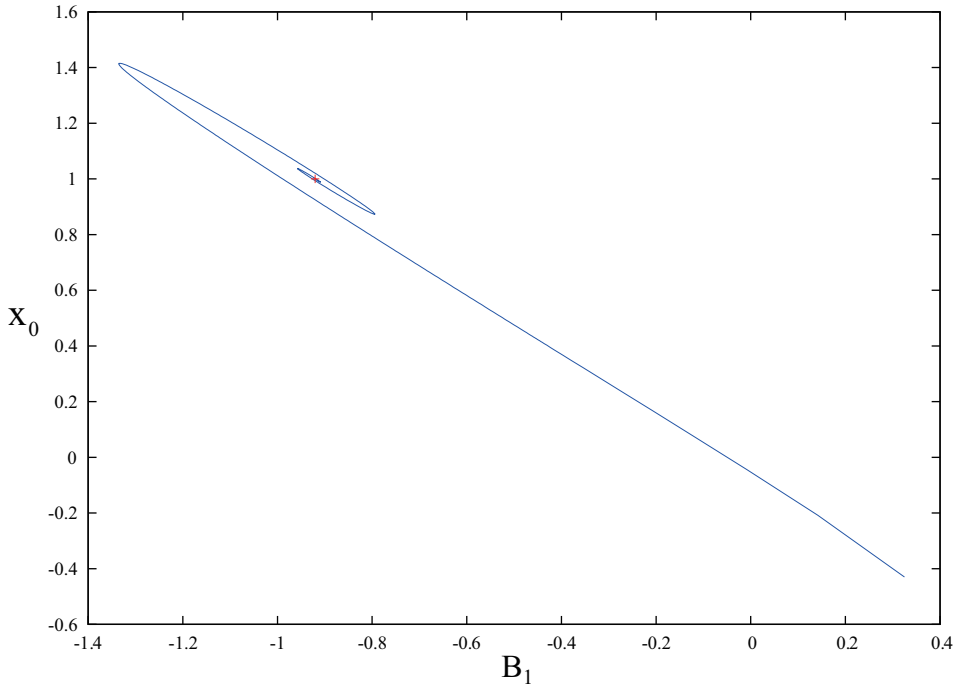


Figure 3.4: Fixed points for the original Alpazur oscillator ($B_2 = 1$). The limit point at the center of the spiral is materialized by a red dot (explanations about this structure in sub-section 3.2.4).

3.1.4 Bifurcation points

The characteristic equation is simple:

$$\chi(\mu) = \det(DT_l - \mu) = 0, \tag{3.14}$$

hence the Jacobian matrix:

$$\begin{bmatrix} \dfrac{\partial x_2}{\partial x_0} - 1 & \dfrac{\partial x_2}{\partial \lambda} \\ \dfrac{\partial DT_l}{\partial x_0} & \dfrac{\partial DT_l}{\partial \lambda} \end{bmatrix} \tag{3.15}$$

Given the projection $U = x$, we can write $DT_l = \partial x_2/\partial x_0$, hence $\partial DT_l/\partial x_0 = \partial^2 x_2/\partial x_0^2$. For this analysis, we have chosen λ as either B_1 or B_2.
We compute the Jacobian elements using a numerical differentiation, as explained in

section 2.4 and 4.1:

$$\begin{aligned}
\frac{\partial x_2}{\partial x_0} &\approx \frac{x_2(x_0+\Delta x,\lambda)-x_2(x_0,\lambda)}{\Delta x} \\
\frac{\partial x_2}{\partial \lambda} &\approx \frac{x_2(x_0,\lambda+\Delta\lambda)-x_2(x_0,\lambda)}{\Delta\lambda} \\
\frac{\partial DT_l}{\partial x_0} &\approx \frac{\frac{\partial x_2}{\partial x_0}(x_0+\Delta x,\lambda)-\frac{\partial x_2}{\partial x_0}(x_0,\lambda)}{\Delta x} \\
\frac{\partial DT_l}{\partial \lambda} &\approx \frac{\frac{\partial x_2}{\partial x_0}(x_0,\lambda+\Delta\lambda)-\frac{\partial x_2}{\partial x_0}(x_0,\lambda)}{\Delta\lambda}.
\end{aligned} \tag{3.16}$$

In our case, we have computed a large number of fixed points and their associated eigenvalues. By simply picking the ones close to a bifurcation point and refining them by applying Newton's method, we obtain a number of bifurcation points for a fixed parameter value (in Fig.3.4, B_2 is fixed).

We then used these points as origin of the bifurcation curves we wanted to compute, applying minute variations to B_2 and correcting x_0 and B_1.

Doing so, we obtained the bifurcation diagram shown in Fig. 3.5.

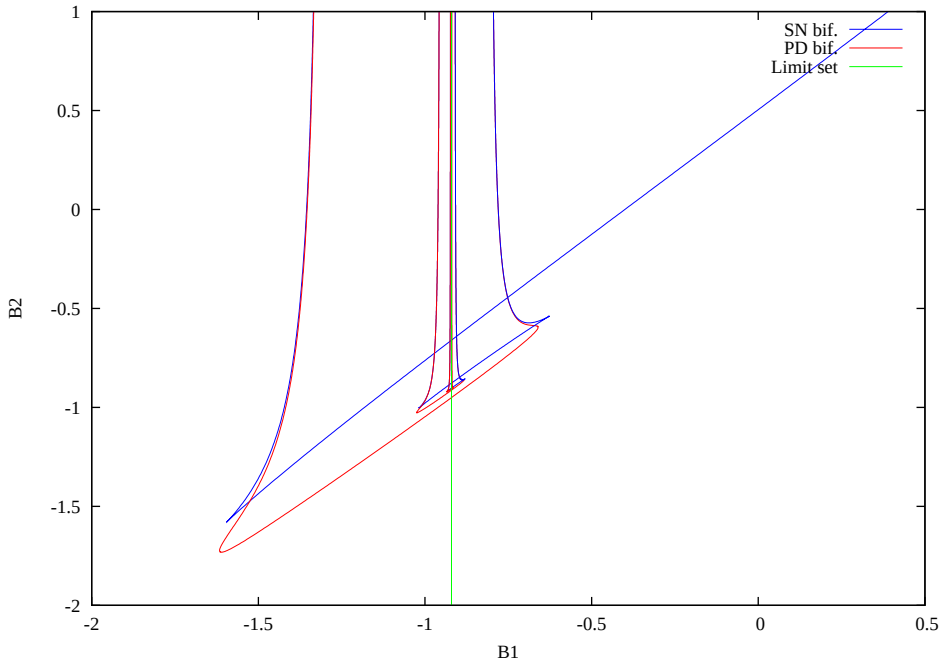


Figure 3.5: Original Alpazur oscillator bifurcation diagram.

The results match the ones available in [9], where an application to collision bifurcation analysis, based on this method, is also detailed. Collision bifurcations occur

when the trajectory of the solution collides with one of the switching lines, under the influence of a parameter change. More considerations about collision bifurcations and their analysis method can be found in the work of Banerjee and al. [4].

Our objective is not to conduct an extensive analysis of the Alpazur oscillator, so the orbits considered are only of one type: $(1, 1)$ periodic orbits, for one sequence per period, and one period for the considered fixed orbit. The definition of such (m, n) orbits has been detailed by Peterka in [19].

3.2 3-state Alpazur oscillator

This is the first extension of the Alpazur oscillator we have studied. The purpose of such analysis is evaluating the influence of the number and quality of switching conditions, and also to confirm the relevance and flexibility of our method in such cases.

3.2.1 Model description

We extend the original Alpazur oscillator from [8] to 3-state (see Fig. 3.6) for examination of the computing method proposed in [9] from a universality point of view, while demonstrating the efficiency of the differentiation approach we used (quickly presented in (3.29), but further detailed in section 4.1).

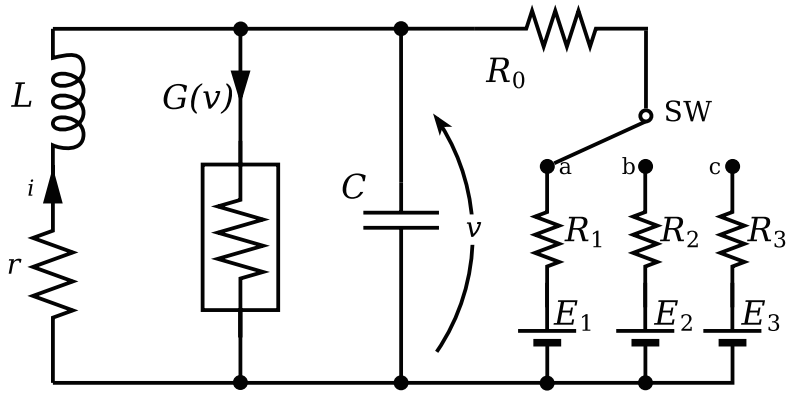


Figure 3.6: Electronic implementation of the 3-state Alpazur oscillator

Still according to the position of the switch, we have the following differential equations:
$f_i(X) = \begin{pmatrix} f_i(x,y) \\ g_i(x,y) \end{pmatrix}, i = 1, 2, 3.$
For state 1: (terminal a in SW on Fig. 3.6)

$$\begin{cases} \dfrac{dx}{dt} = f_1(x,y) = -rx - y \\ \dfrac{dy}{dt} = g_1(x,y) = x + (1 - A_1)y - \dfrac{1}{3}y^3 + B_1. \end{cases} \tag{3.17}$$

For state 2: (terminal b in SW on Fig. 3.6)

$$\begin{cases} \dfrac{dx}{dt} = f_2(x,y) = -rx - y \\ \dfrac{dy}{dt} = g_2(x,y) = x + (1 - A_2)y - \dfrac{1}{3}y^3 + B_2. \end{cases} \tag{3.18}$$

For state 3: (terminal c in SW on Fig. 3.6)

$$\begin{cases} \dfrac{dx}{dt} = f_3(x,y) = -rx - y \\ \dfrac{dy}{dt} = g_3(x,y) = x + (1 - A_3)y - \dfrac{1}{3}y^3 + B_3. \end{cases} \tag{3.19}$$

The switching rules are:

$$\begin{aligned} q_1(x,y) &= y - h \\ q_2(x,y) &= y - b \\ q_3(x,y) &= y - m, \end{aligned} \tag{3.20}$$

with a fixed switching sequence for a period: state 1, state 2, state 3.

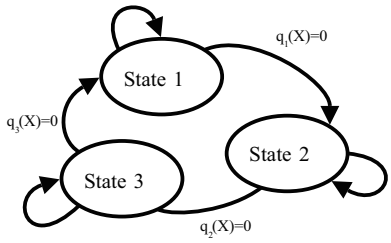


Figure 3.7: Petri diagram of the 3-state Alpazur oscillator discrete sequence.

This results in phase portraits in the hybrid space (state-associated planes as shown in Fig. 3.8.)

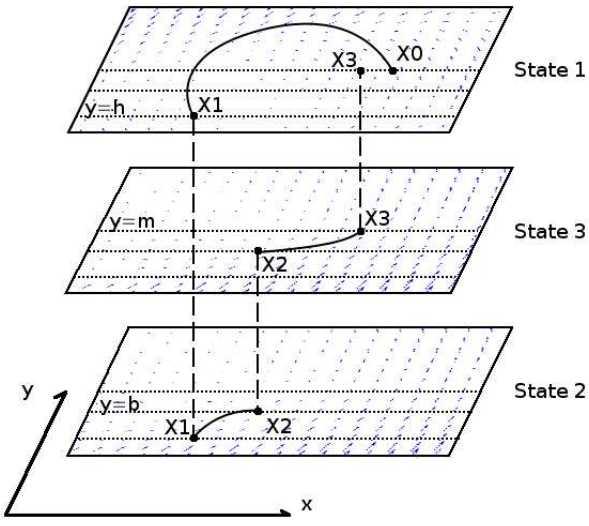


Figure 3.8: Hysteresis of the switching constraints of Alpazur oscillator

The resulting system exhibits chaotic orbits at particular parameter values, as shown

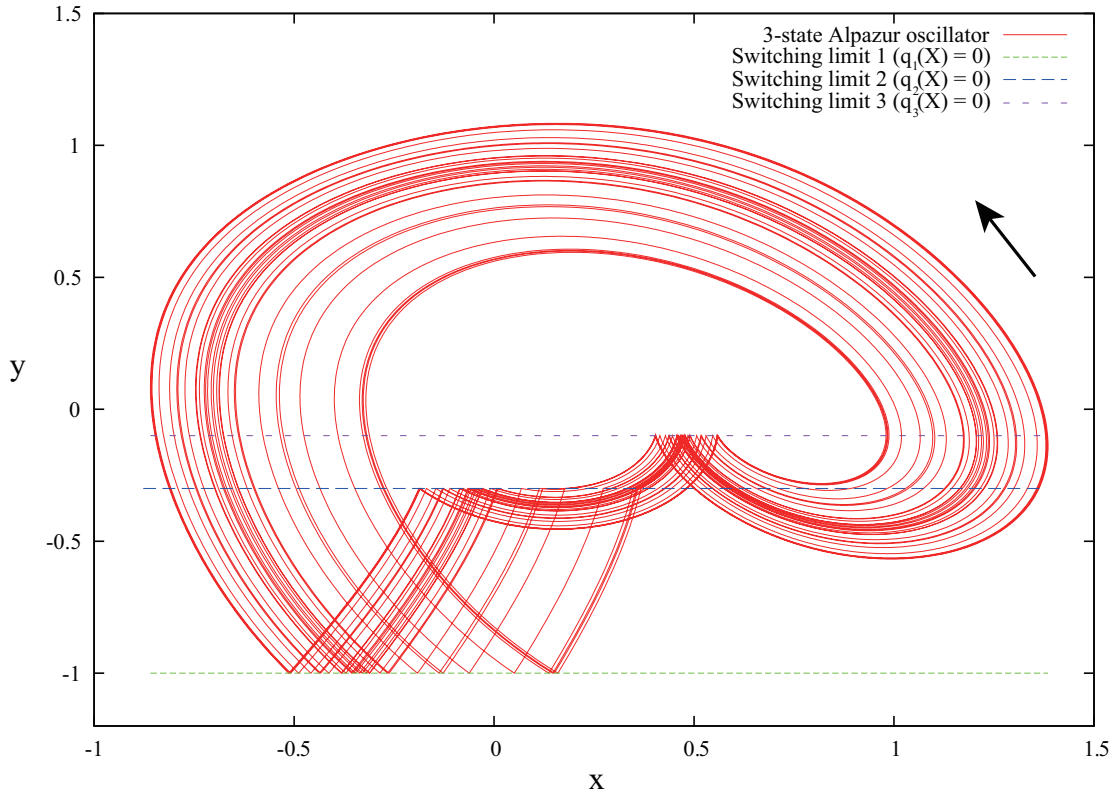


Figure 3.9: Sample phase portrait of chaotic behavior

in Fig. 3.9, where we used the following set of parameters:
$r = 0.1 \quad A_1 = 0.2 \quad A_2 = 2.0 \quad A_3 = 0.8 \quad B_1 = -0.6$
$B_2 = 1.0 \quad B_3 = -0.1 \quad m = -0.1 \quad b = -0.3 \quad h = -1.0$

This model is convenient because the mapping is straight forward: even within the local maps, y values are fixed due to the switching conditions. Therefore, we can extract the mapped variable $U = x$. We define the map:

$$\begin{array}{lcccl}
T_1 & : & \Pi_0 & \to & \Pi_1 \\
 & & x_0 & \mapsto & x_1 = \varphi_1(\tau_0, x_0, y_0) \\
 & & y_0 & \mapsto & y_1 = \phi_1(\tau_0, x_0, y_0) = h \\
T_2 & : & \Pi_1 & \to & \Pi_2 \\
 & & x_1 & \mapsto & x_2 = \varphi_2(\tau_1, x_1, y_1) \\
 & & y_1 & \mapsto & y_2 = \phi_2(\tau_1, x_1, y_1) = b \\
T_3 & : & \Pi_2 & \to & \Pi_0 \\
 & & x_2 & \mapsto & x_3 = \varphi_3(\tau_2, x_2, y_2) \\
 & & y_2 & \mapsto & y_3 = \phi_3(\tau_2, x_2, y_2) = m \\
T = T_3 \circ T_2 \circ T_1 & : & \Pi_0 & \to & \Pi_0 \\
 & & x_0 & \mapsto & x_3 = \varphi(\tau, x_0).
\end{array} \tag{3.21}$$

3.2.2 Fixed points

The problem of fixed points corresponds to finding x_0 so that:

$$x_3 - x_0 = 0. \tag{3.22}$$

In order to achieve the appropriate correction, we need to compute:

$$DT_l = \frac{dx_3}{dx_0} = \frac{dx_3}{dx_2}\frac{dx_2}{dx_1}\frac{dx_1}{dx_0}. \tag{3.23}$$

For each State i we compute:

$$\begin{aligned} \frac{dx_i}{dx_{i-1}} &= \frac{\partial \varphi_i}{\partial x_{i-1}} + f_i(x_i, y_i)\frac{\partial \tau_i}{\partial x_{i-1}} \\ \frac{\partial \tau_i}{\partial x_{i-1}} &= \frac{-\dfrac{\partial \phi_i}{\partial x_{i-1}}}{g_i(x_i, y_i)}, \end{aligned} \tag{3.24}$$

where we numerically integrate the required elements:

$$\begin{aligned} &\text{State 1: from } \begin{bmatrix} x_0 \\ m \end{bmatrix} \\ \frac{d}{dt}\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i(x,y) \\ g_i(x,y) \end{bmatrix} &\quad \text{State 2: from } \begin{bmatrix} x_1 \\ h \end{bmatrix} \\ &\text{State 3: from } \begin{bmatrix} x_2 \\ b \end{bmatrix} \\ \frac{d}{dt}\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix} &= \begin{bmatrix} \dfrac{\partial f_i}{\partial x} & \dfrac{\partial f_i}{\partial y} \\ \dfrac{\partial g_i}{\partial x} & \dfrac{\partial g_i}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix} \\ \text{where } \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} \end{bmatrix}_{t=0} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \end{aligned} \tag{3.25}$$

Note that we could drop $\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial \phi_i}{\partial y_{i-1}} \end{bmatrix}$ since y_i values are fixed.

We now use the Newton method to compute the next x_0 value:

$$x_0' = x_0 - \frac{x_3 - x_0}{\dfrac{dx_3}{dx_0} - 1}. \tag{3.26}$$

3.2.3 Bifurcation points

The characteristic equation is:

$$\chi(\mu) = \det(DT_l - \mu) = 0, \tag{3.27}$$

hence the Jacobian matrix:

$$\begin{bmatrix} \dfrac{\partial x_3}{\partial x_0} - 1 & \dfrac{\partial x_3}{\partial \lambda} \\ \dfrac{\partial DT_l}{\partial x_0} & \dfrac{\partial DT_l}{\partial \lambda} \end{bmatrix} \tag{3.28}$$

$DT_l = \partial x_3/\partial x_0$, hence $\partial DT_l/\partial x_0 = \partial^2 x_3/\partial x_0^2$. λ still corresponds to B_1 or B_2.
We compute the Jacobian elements:

$$\begin{aligned} \frac{\partial x_3}{\partial x_0} &\approx \frac{x_3(x_0 + \Delta x, \lambda) - x_3(x_0, \lambda)}{\Delta x} \\ \frac{\partial x_3}{\partial \lambda} &\approx \frac{x_3(x_0, \lambda + \Delta\lambda) - x_3(x_0, \lambda)}{\Delta\lambda} \\ \frac{\partial DT_l}{\partial x_0} &\approx \frac{\frac{\partial x_3}{\partial x_0}(x_0 + \Delta x, \lambda) - \frac{\partial x_3}{\partial x_0}(x_0, \lambda)}{\Delta x} \\ \frac{\partial DT_l}{\partial \lambda} &\approx \frac{\frac{\partial x_3}{\partial x_0}(x_0, \lambda + \Delta\lambda) - \frac{\partial x_3}{\partial x_0}(x_0, \lambda)}{\Delta\lambda}. \end{aligned} \tag{3.29}$$

Then, we obtain the bifurcation diagram seen in Fig. 3.10.

3.2.4 Results review

By looking at the bifurcation diagram Fig.3.10, we can see a redundant bifurcation pattern (enlargement in Fig.3.11) in the neighborhood of a particular value of B_1 (to which we will refer as the limit value B_1^*).
Due to precision limitations inherent to numerical methods, it gets more and more difficult to compute the bifurcation lines as we get closer to the limit value, but we can conjecture this structure is infinite.

Such a bifurcation structure has been previously identified and studied by Carcasses & Mira [14], in a 3-dimensional linear system. In the case of the considered linear system, it has been proved that cusp points exist in the the parameter plane. The existence of cusp points in the parameter plane is related to the existence of specific structures such as dovetail structure, spring and cross-road areas [14]. The switching characteristic of the system makes possible for the orbit to return to an unstable slow motion region after a state switch.
To understand this structure, we will consider various results:
First, let us fix a value for B_2, and compute a fixed points line γ in the B_1 / x_0 plane (see Fig.3.12).

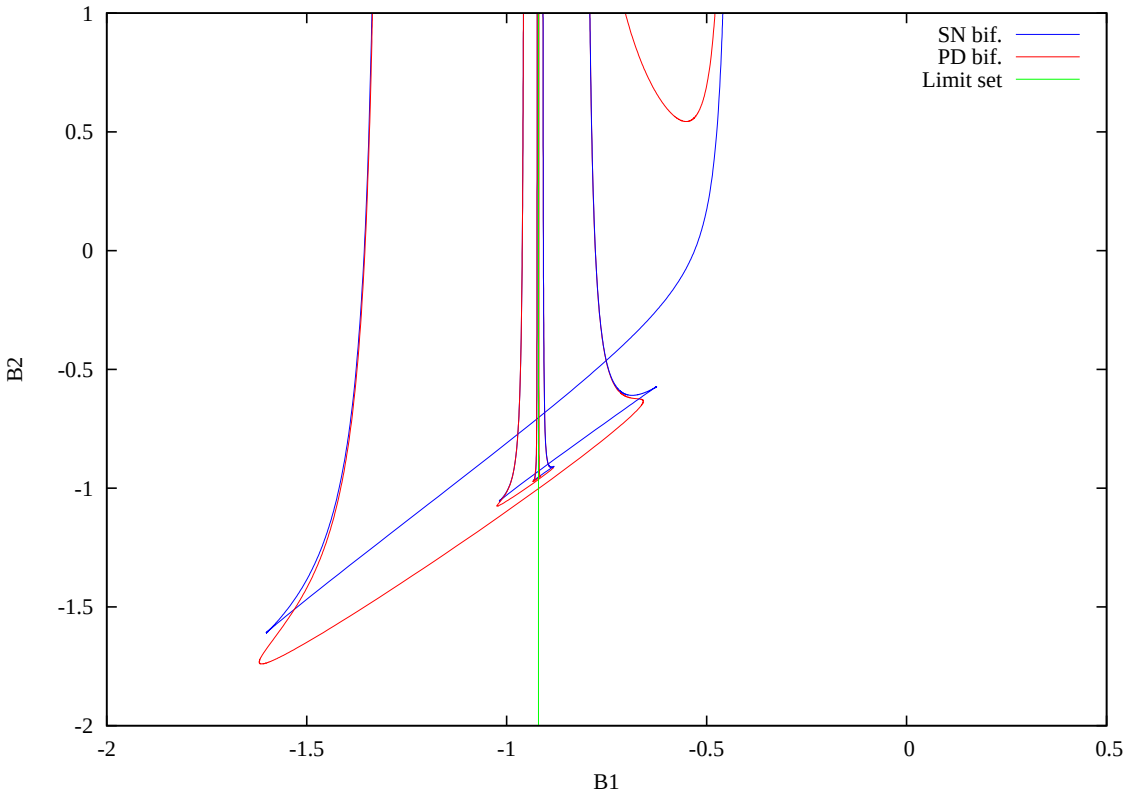


Figure 3.10: 3-state Alpazur oscillator bifurcation diagram.

If we compute the phase portraits along this curve (visible in Fig.3.13), we can see that the solution is gradually wrapping itself around an equilibrium point at state 1: $f_1(X) = 0$.
We can see what is referred to by Carcasses & Mira in [14] as "slow motions part of the trajectory" in the phase portraits, with a limit: the center of the spiral in Fig. 3.12. At this point, the equilibrium point is on the initial switching line, and the initial point (X_0) merges with this equilibrium point, leading to these slow motions.
It results in the following equation set:

$$\begin{cases} f_1(X_0) = \begin{bmatrix} -rx_0 - y_0 \\ x_0 + (1 - A_1)y_0 - \frac{1}{3}y_0^3 + B_1^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ y_0 = m \end{cases} \tag{3.30}$$

where r, A_1, and m are known fixed parameters; in our case $r = 0.1$, $A_1 = 0.2$ and $m = -0.1$.
Thus we obtain:

$$\begin{cases} x_0 = -\dfrac{m}{r} = 1 \\ B_1^* = m\left(\dfrac{1}{r} - 1 + A_1 + \dfrac{m^2}{3}\right) = -\dfrac{2761}{3000} \end{cases} \tag{3.31}$$

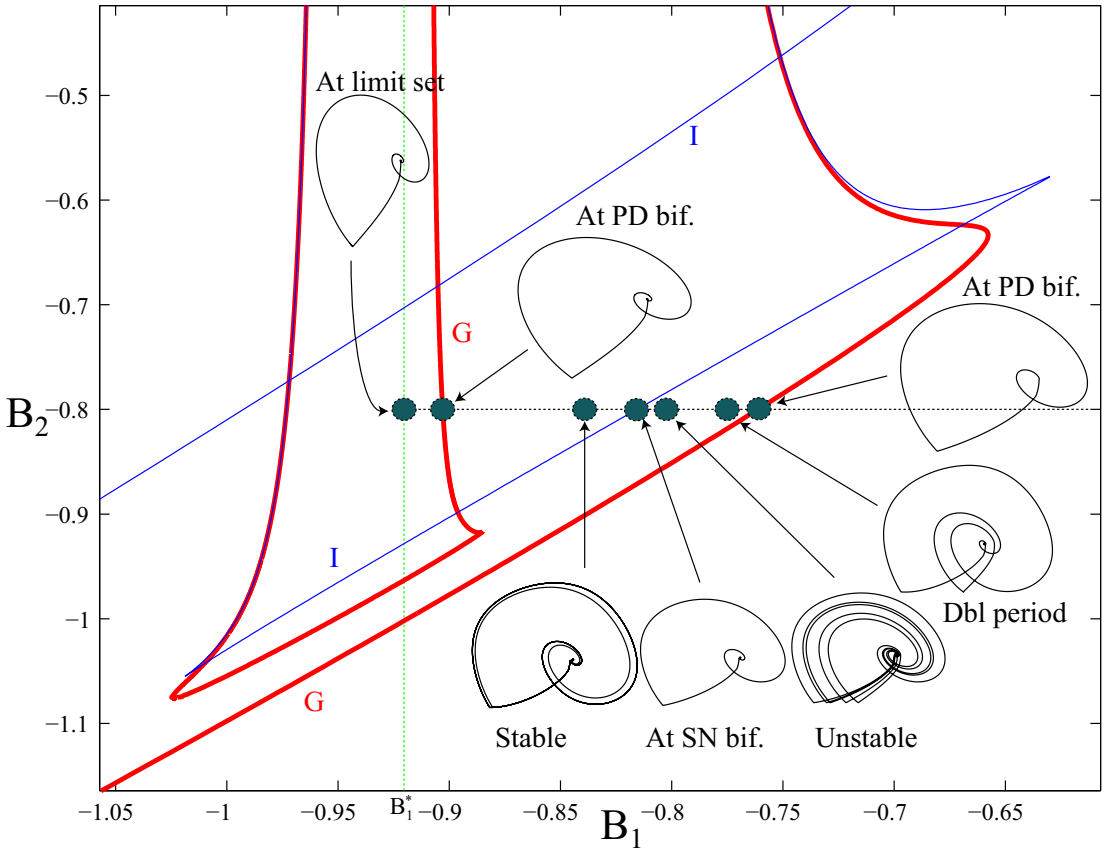


Figure 3.11: Enlargement of the 3-state Alpazur oscillator bifurcation diagram around the limit set area.

If we try to compute the limit phase portrait using such values, we fail because the initial point is an equilibrium point. Instead, we used reversed-time to compute the limit phase portrait at an arbitrary value of B_2 (see Fig. 3.14.)
If we monitor the characteristic (3.27) along the curve Fig. 3.12, we find eigenvalues corresponding to local bifurcations. For instance, we can confirm that each point where the curve's tangent verifies $\partial\gamma/\partial B_1 = 0$, is a saddle-node bifurcation point, close to which we can find a period doubling bifurcation point. Starting from those bifurcation points, we compute bifurcation lines in the B_1 / B_2 plane, resulting in the structure in Fig. 3.10. From what we have seen of the bifurcations around the limit set defined by $B_1 = B_1^*$, we can conjecture that for each bifurcation line, we can find another one closer to this limit set. This is visible if we mix Fig. 3.12 with the bifurcation diagram Fig. 3.10 as in Fig. 3.15 (also illustrated in Fig. 3.16 for better visibility), where we can associate the infinite amount of loops in the spiral of γ to the infinite number of bifurcation lines.
As for the cusps on the saddle-node bifurcation lines, we can compute precisely their

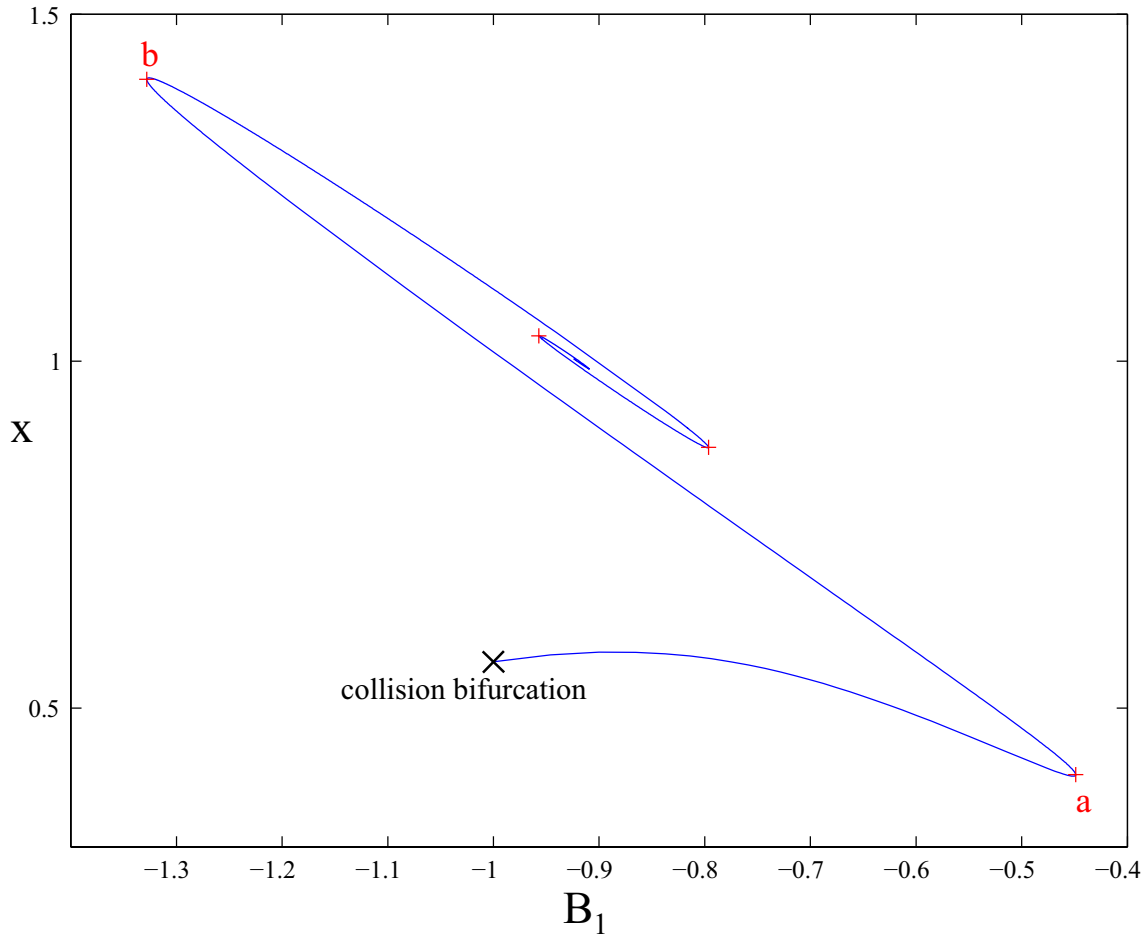


Figure 3.12: Fixed points line γ at an arbitrary value $B_2 = 1.92$, in the plane B_1 / x_0. We can see the fixed points curve forms a spiral.

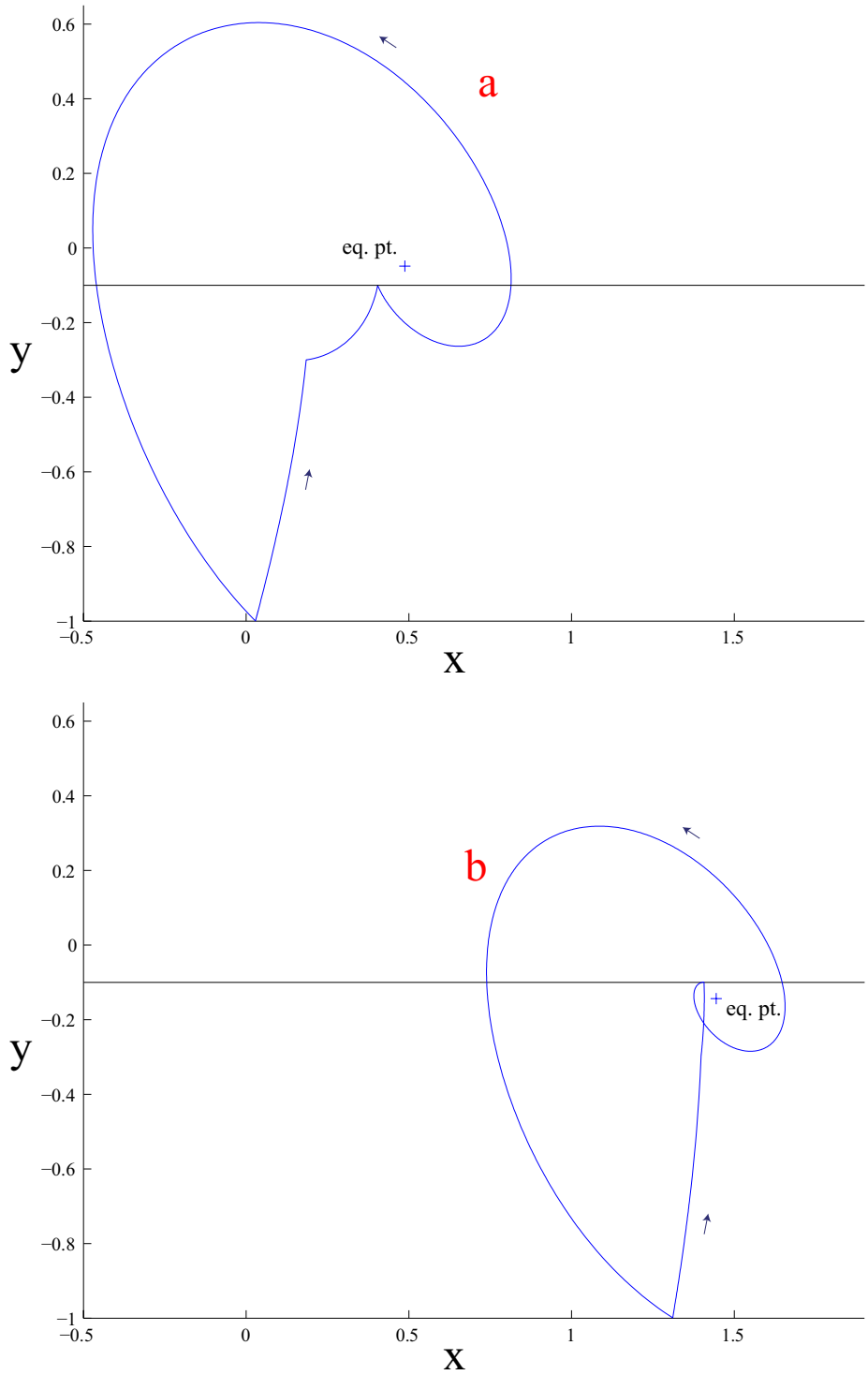


Figure 3.13: Transition between fixed points with eigenvalues corresponding to saddle-node bifurcation (see Fig.3.12). Each loop in the spiral corresponds to an extra loop around the equilibrium point in the phase portraits.

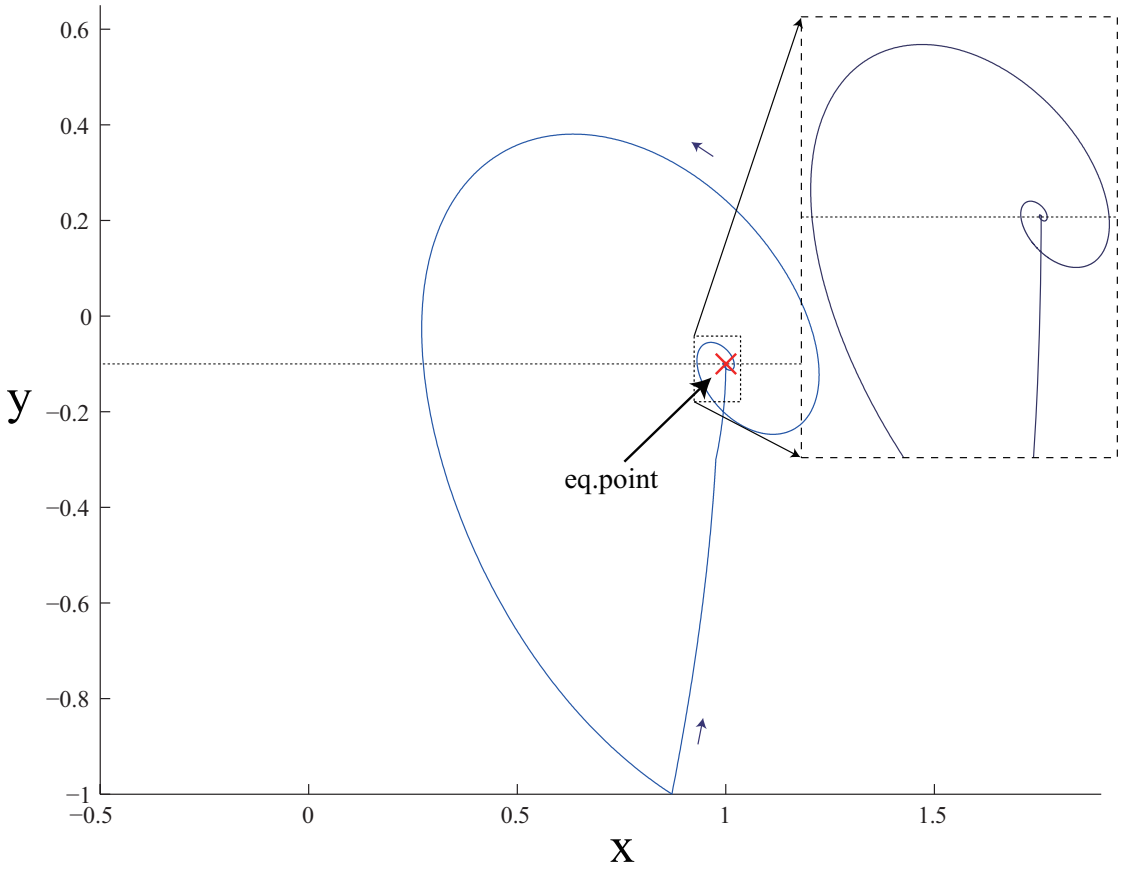


Figure 3.14: Phase portrait of a fixed point on the limit set $B_1 = B_1^*$. This is actually a limit orbit computed using reversed time, since the initial point is an unstable equilibrium point at state 1. The resulting focus is tending to the equilibrium point in such slow motion area , suggesting a period $\tau \to \infty$.

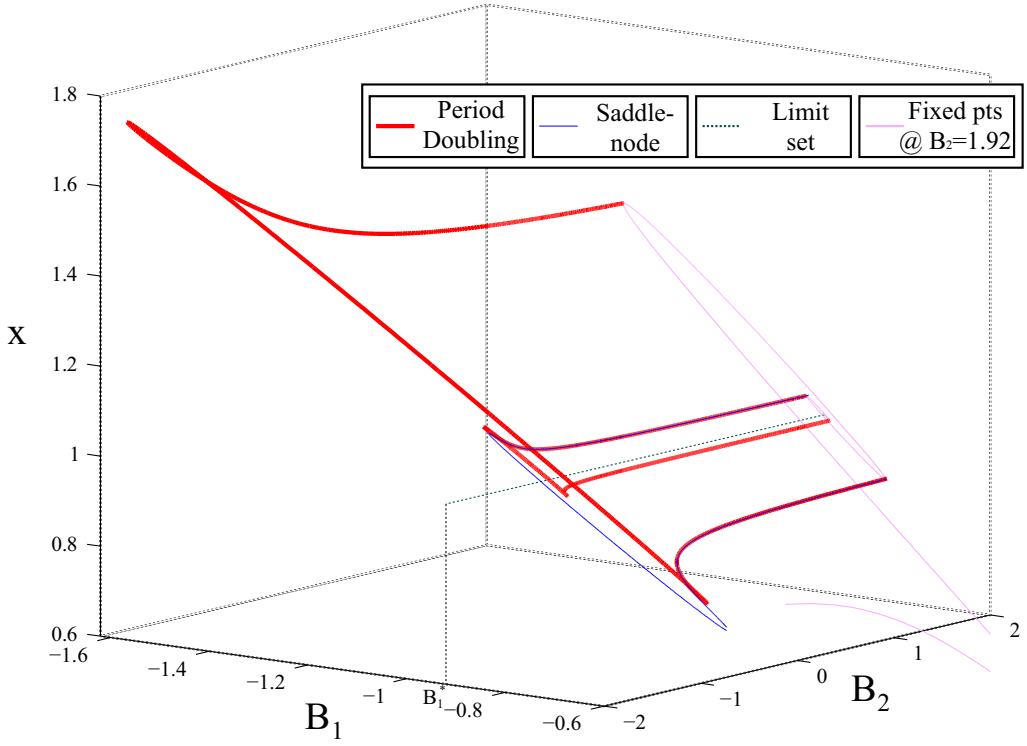


Figure 3.15: Bifurcation diagram put against the fixed points set at $B_2 = 1.92$

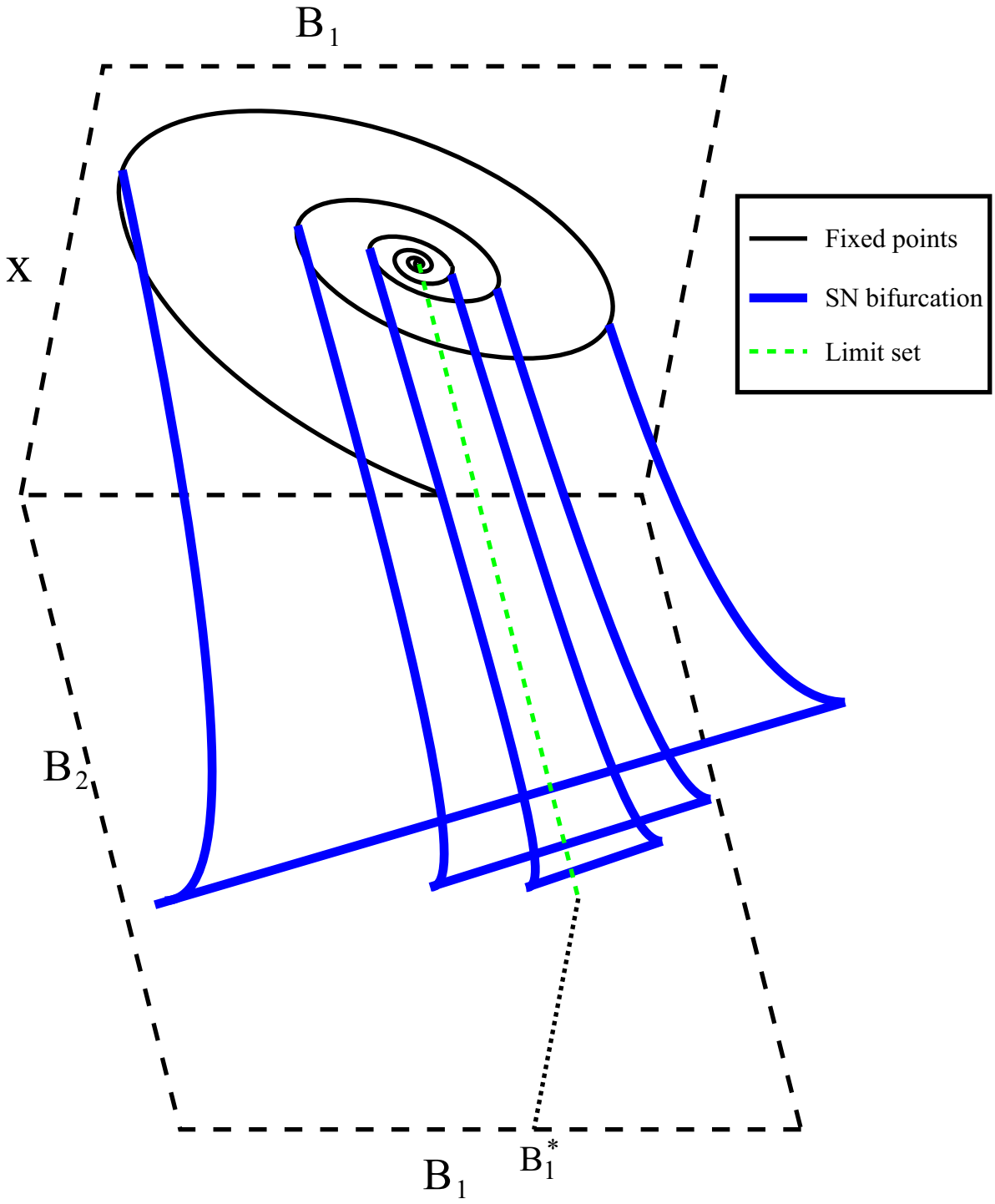


Figure 3.16: Schematic view of Fig.3.15.

position by solving the following:

$$\begin{cases} \dfrac{\partial x_3}{\partial x_0} = 1 \\ \dfrac{\partial DT_l}{\partial B_1} = 0 \\ \dfrac{\partial DT_l}{\partial B_2} = 0 \end{cases} \tag{3.32}$$

We can find a useful method to compute the bifurcation lines through cusps flawlessly proposed by Kitajima and Kawakami in [21]. In the scope of cusp generation pattern analysis, it appears possible to take advantage of the structure's relative monotony, and use a linearized model as did Carcasses & Mira [14].

The fact that this bifurcation structure seems to be more dependent from the interactions between equilibrium points and switching limits rather than the nature of the switching conditions means that we can expect to find this kind of structure in bifurcation diagrams of other circuits if we carefully choose parameters. It also reveals that, if we want to study this type of bifurcation structure more into details, linear switching conditions appear sufficient.

3.3 Varieties of switching thresholds

The next step in testing out analysis method is to change the switching conditions, not in quantity but in quality. As we study a variety of switching conditions, we can see how the Poincaré map helps us handling various system scenarios.

3.3.1 2-state Alpazur oscillator with affine switching condition

This is the same oscillator as the original 2-state version but we modified the switching conditions:

$$\begin{aligned} q_1(x, y) &= y + 1.0 - 0.2x \\ q_2(x, y) &= y + 0.1, \end{aligned} \tag{3.33}$$

Note that the first switching condition is an affine function of both x and y.

This system exhibits chaotic orbits at parameter values such as:
$r = 0.1 \quad A_1 = 0.2 \quad A_2 = 2.0 \quad B_1 = -0.2 \quad B_2 = 1.0$

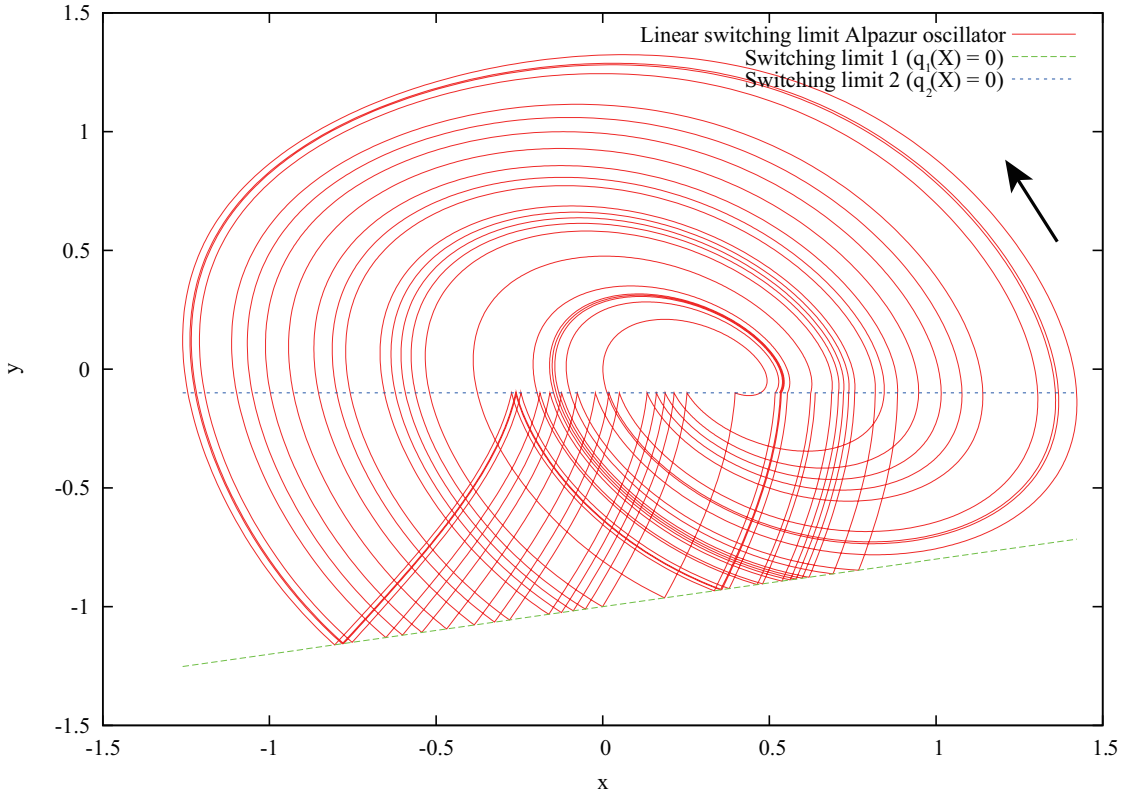


Figure 3.17: Sample phase portrait of chaotic behavior

This time, y_1 do vary, which mean we have to take it into account:

$$\begin{aligned} DT_l &= \frac{\partial x_2}{\partial x_0} \\ &= \frac{\partial x_2}{\partial X_1}\frac{\partial X_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} \\ &= \frac{\partial x_2}{\partial x_1}\frac{\partial x_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} + \frac{\partial x_2}{\partial y_1}\frac{\partial y_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} \\ &= \frac{\partial x_2}{\partial x_1}\frac{\partial x_1}{\partial x_0} + \frac{\partial x_2}{\partial y_1}\frac{\partial y_1}{\partial x_0}. \end{aligned} \tag{3.34}$$

For each State i we compute:

$$\begin{cases} \dfrac{\partial x_i}{\partial x_{i-1}} &= \dfrac{\partial \varphi_i}{\partial x_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial x_{i-1}} \\ \dfrac{\partial \tau_i}{\partial x_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial x_{i-1}}q_{iy} - \dfrac{\partial \phi_i}{\partial x_{i-1}}q_{ix}}{f_i(x_i, y_i)q_{iy} - g_i(x_i, y_i)q_{ix}} \end{cases} \tag{3.35}$$

$$\begin{cases} \dfrac{\partial x_i}{\partial y_{i-1}} &= \dfrac{\partial \varphi_i}{\partial y_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial y_{i-1}} \\ \dfrac{\partial \tau_i}{\partial y_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial y_{i-1}}q_{iy} - \dfrac{\partial \phi_i}{\partial y_{i-1}}q_{ix}}{f_i(x_i, y_i)q_{iy} - g_i(x_i, y_i)q_{ix}} \end{cases}, \tag{3.36}$$

where $\partial y_i/\partial x_i = \partial y_i/\partial x|_{q_i(x_i,y_i)=0 \text{ and } x=x_i}$; q_{ix} and q_{iy} are the components of a vector tangent to the switching curve at X_i.
Then there are two ways of calculating $\partial y_1/\partial x_0$:

$$\begin{aligned} \frac{\partial y_1}{\partial x_0} &= \frac{\partial \phi_1}{\partial x_0} + g_1(x_1, y_1)\frac{\partial \tau_1}{\partial x_0} \\ &= \frac{\partial y_1}{\partial x_1}\frac{\partial x_1}{\partial x_0} \end{aligned} \tag{3.37}$$

We numerically integrate the required elements:

$$\begin{aligned} &\frac{d}{dt}\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i(x,y) \\ g_i(x,y) \end{bmatrix} \quad \text{State 1: from } x_0 \\ &\quad \text{State 2: from } x_1 \\ &\frac{d}{dt}\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} & \dfrac{\partial \varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} & \dfrac{\partial \phi_i}{\partial y_{i-1}} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_i}{\partial x} & \dfrac{\partial f_i}{\partial y} \\ \dfrac{\partial g_i}{\partial x} & \dfrac{\partial g_i}{\partial y} \end{bmatrix}\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} & \dfrac{\partial \varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} & \dfrac{\partial \phi_i}{\partial y_{i-1}} \end{bmatrix} \\ &\text{where } \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial x_{i-1}} & \dfrac{\partial \varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial \phi_i}{\partial x_{i-1}} & \dfrac{\partial \phi_i}{\partial y_{i-1}} \end{bmatrix}_{t=0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.38}$$

The Newton method can then be applied the same way as before to compute the correction.

Computing the Jacobian matrix with the numerical method we describe in section 4.1 greatly simplifies the process as there is no need to take into account the change of switching condition any further. Then, we obtain the bifurcation diagram in Fig. 3.18.

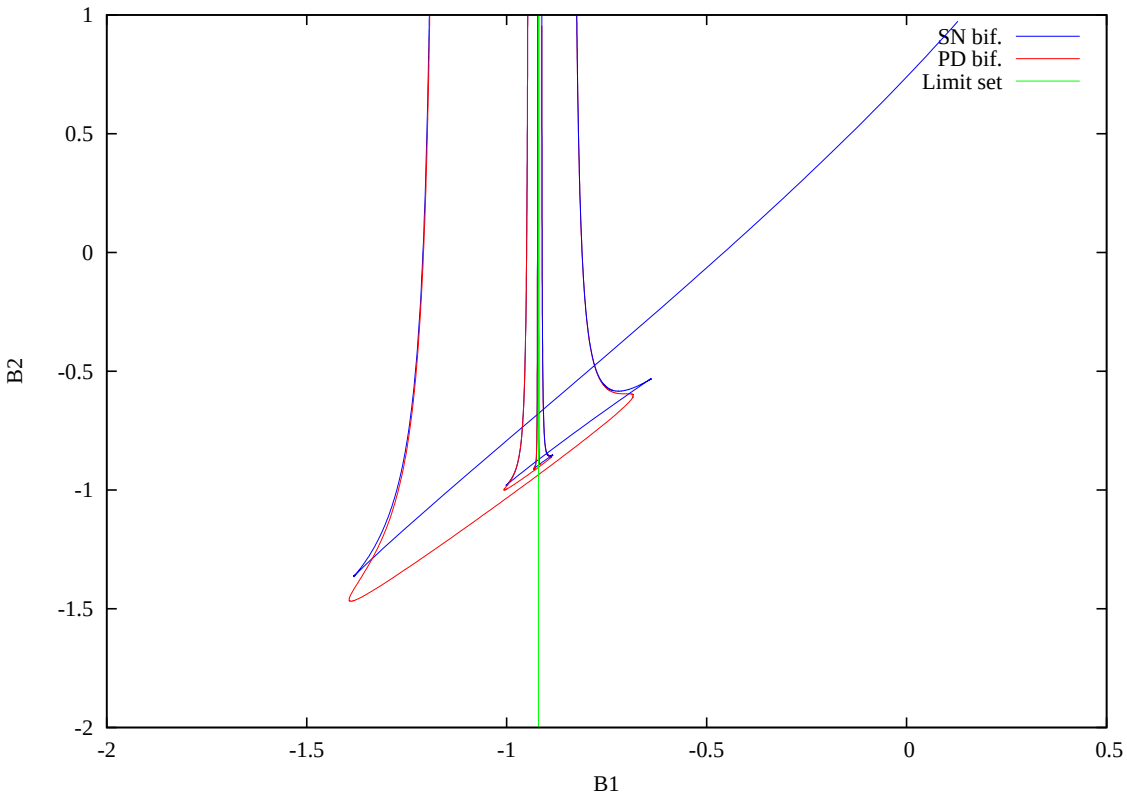


Figure 3.18: 2-state Alpazur oscillator with affine switching condition bifurcation diagram.

3.3.2 2-state Alpazur oscillator with nonlinear switching condition

We still base this system on the original 2-state Alpazur oscillator, only this time we will use nonlinear switching conditions:

$$\begin{aligned} q_1(x,y) &= y + 1.0 - 0.2\sin x \\ q_2(x,y) &= y + 0.1 - 0.2x^2, \end{aligned} \tag{3.39}$$

Such switching conditions are indeed very unlikely, but they demonstrate the efficiency of the method even for complex switching cases. This system exhibits chaotic orbits at similar parameter values as with affine switching limits (see Fig. 3.19.)

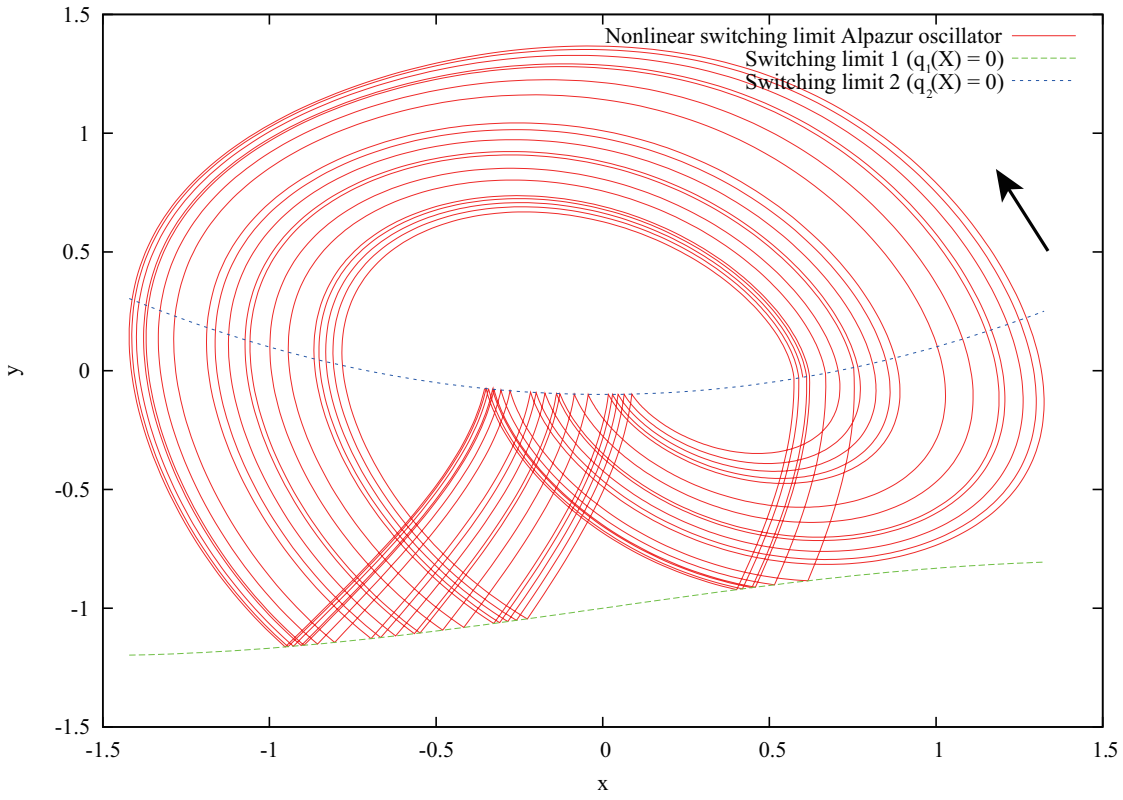


Figure 3.19: Sample phase portrait of chaotic behavior

A change in the switching limit leads to a new expression of DT_l:

$$\begin{aligned} DT_l &= \frac{\partial x_2}{\partial x_0} \\ &= \frac{\partial x_2}{\partial X_1}\frac{\partial X_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} \\ &= \frac{\partial x_2}{\partial x_1}\frac{\partial x_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} + \frac{\partial x_2}{\partial y_1}\frac{\partial y_1}{\partial X_0}\frac{\partial X_0}{\partial x_0} \\ &= \frac{\partial x_2}{\partial x_1}\left(\frac{\partial x_1}{\partial x_0} + \frac{\partial x_1}{\partial y_0}\frac{\partial y_0}{\partial x_0}\right) + \frac{\partial x_2}{\partial y_1}\left(\frac{\partial y_1}{\partial x_0} + \frac{\partial y_1}{\partial y_0}\frac{\partial y_0}{\partial x_0}\right). \end{aligned} \tag{3.40}$$

For each state i we compute:

$$\begin{cases} \dfrac{\partial x_i}{\partial x_{i-1}} &= \dfrac{\partial \varphi_i}{\partial x_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial x_{i-1}} \\ \dfrac{\partial \tau_i}{\partial x_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial x_{i-1}}q_{iy} - \dfrac{\partial \phi_i}{\partial x_{i-1}}q_{ix}}{f_i(x_i, y_i)q_{iy} - g_i(x_i, y_i)q_{ix}} \end{cases} \tag{3.41}$$

$$\begin{cases} \dfrac{\partial x_i}{\partial y_{i-1}} &= \dfrac{\partial \varphi_i}{\partial y_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial y_{i-1}} \\ \dfrac{\partial \tau_i}{\partial y_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial y_{i-1}}q_{iy} - \dfrac{\partial \phi_i}{\partial y_{i-1}}q_{ix}}{f_i(x_i, y_i)q_{iy} - g_i(x_i, y_i)q_{ix}} \end{cases}, \tag{3.42}$$

where $\partial y_i/\partial x_i = \partial y_i/\partial x|_{q_i(x_i,y_i)=0 \text{ and } x=x_i}$; q_{ix} and q_{iy} are the components of a vector tangent to the switching curve at X_i.
Then there are two ways of calculating $\partial y_i/\partial x_{i-1}$ and $\partial y_i/\partial y_{i-1}$:

$$\begin{aligned} \frac{\partial y_i}{\partial x_{i-1}} &= \frac{\partial \phi_i}{\partial x_{i-1}} + g_i(x_i, y_i)\frac{\partial \tau_i}{\partial x_{i-1}} \\ &= \frac{\partial y_i}{\partial x_i}\frac{\partial x_i}{\partial x_{i-1}} \end{aligned} \tag{3.43}$$

$$\begin{aligned} \frac{\partial y_i}{\partial y_{i-1}} &= \frac{\partial \phi_i}{\partial y_{i-1}} + g_i(x_i, y_i)\frac{\partial \tau_i}{\partial y_{i-1}} \\ &= \frac{\partial y_i}{\partial x_i}\frac{\partial x_i}{\partial y_{i-1}} \end{aligned}. \tag{3.44}$$

We numerically integrate the required elements:

$$\begin{aligned}
&\frac{d}{dt}\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i(x,y) \\ g_i(x,y) \end{bmatrix} \quad \text{State 1: from } x_0 \\
&\frac{d}{dt}\begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_i}{\partial x} & \dfrac{\partial f_i}{\partial y} \\ \dfrac{\partial g_i}{\partial x} & \dfrac{\partial g_i}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix} \quad \text{State 2: from } x_1 \\
&\text{where } \begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix}_{t=0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned} \tag{3.45}$$

As in the previous case, even with complex switching conditions, the numerical differentiation used to obtain second derivatives lifts the need to change the equations any further. At this step, getting an general expression of elements such as $\partial^2 x_2/\partial x_0^2$, $\partial^2 x_2/\partial x_0 \partial\lambda$ and so on would become a real issue. Instead using the same approach as with previous version of Alpazur oscillator, we easily obtain the bifurcation diagram Fig. 3.20.

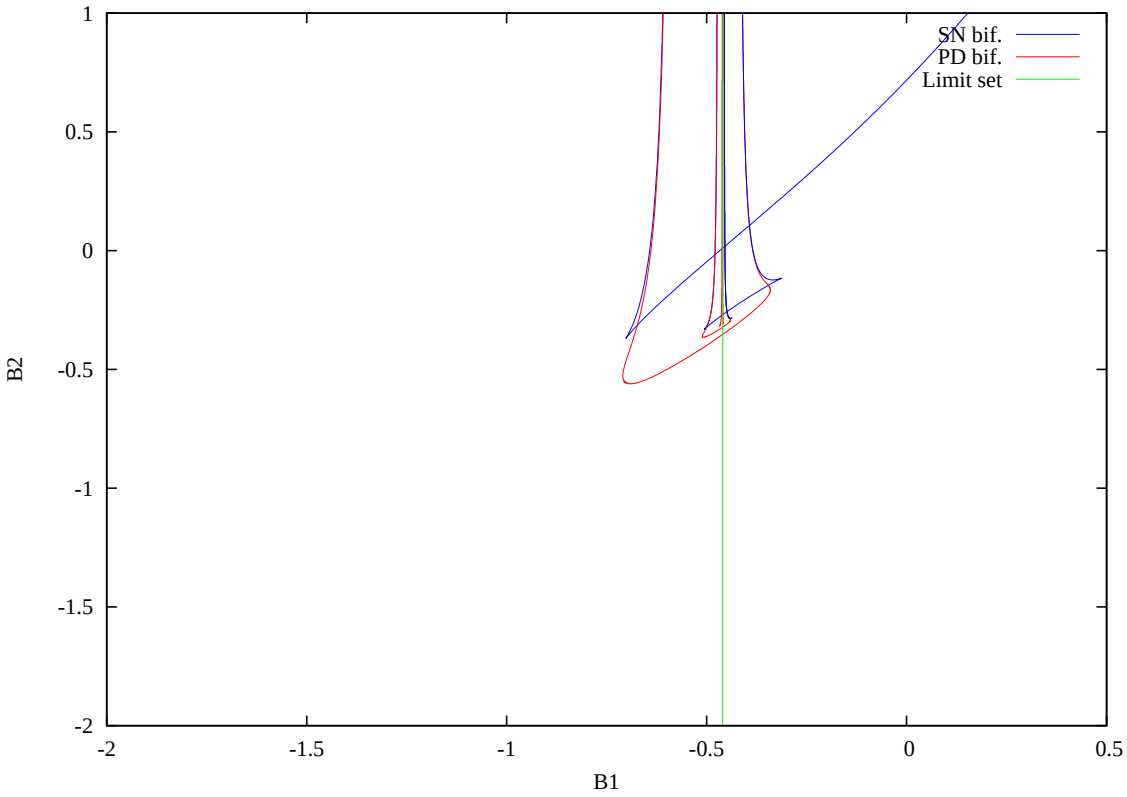


Figure 3.20: 2-state Alpazur oscillator with affine switching condition bifurcation diagram.

3.3.3 2-state Alpazur oscillator with non-smooth switching condition

The switching rules are as follows:

$$\begin{aligned} q_1(x,y) &= y + 1.1 - 0.2x \quad \text{for } x \leq 0.5 \\ q_1(x,y) &= y + 1.0 \quad \text{for } x \geq 0.5 \\ q_2(x,y) &= y + 0.1, \end{aligned} \tag{3.46}$$

This system exhibits chaotic orbits at particular parameter values such as the following set:
$r = 0.1 \quad A_1 = 0.2 \quad A_2 = 2.0 \quad B_1 = -1.0 \quad B_2 = -0.9$
In order to achieve the appropriate correction, we need to compute:

$$DT_l = \frac{\partial x_2}{\partial x_0} = \frac{\partial x_2}{\partial x_1}\frac{\partial x_1}{\partial x_0} + \frac{\partial x_2}{\partial y_1}\frac{\partial y_1}{\partial x_0}. \tag{3.47}$$

This is the same complexity as (3.34). Also note that for $x_1 \geq 0.5$, $\partial y_1/\partial x_0 = 0$ simplifying the process to the analysis of an orbit of the standard Alpazur Oscillator 3.10.

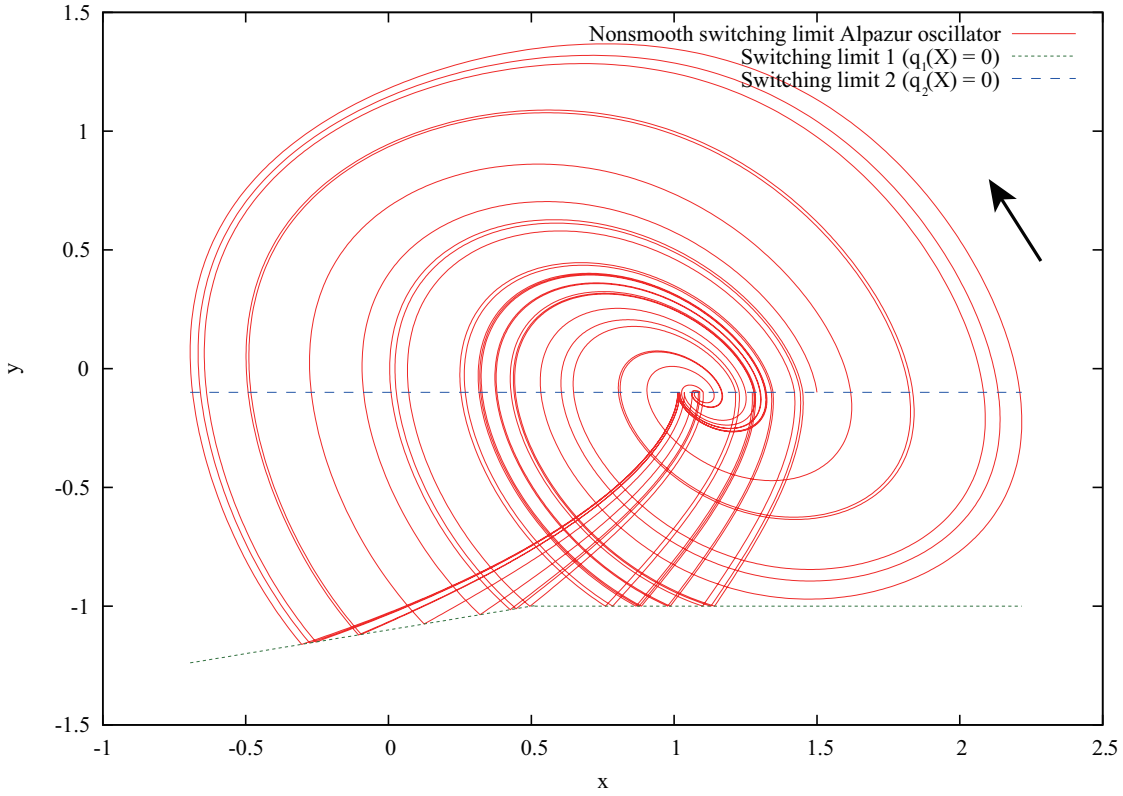


Figure 3.21: Sample phase portrait of chaotic behavior

For each State i we compute:

$$\begin{cases} \dfrac{\partial x_i}{\partial x_{i-1}} &= \dfrac{\partial \varphi_i}{\partial x_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial x_{i-1}} \\ \dfrac{\partial \tau_i}{\partial x_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial x_{i-1}} q_{iy} - \dfrac{\partial \phi_i}{\partial x_{i-1}} q_{ix}}{f_i(x_i, y_i) q_{iy} - g_i(x_i, y_i) q_{ix}} \end{cases} \tag{3.48}$$

$$\begin{cases} \dfrac{\partial x_i}{\partial y_{i-1}} &= \dfrac{\partial \varphi_i}{\partial y_{i-1}} + f_i(x_i, y_i)\dfrac{\partial \tau_i}{\partial y_{i-1}} \\ \dfrac{\partial \tau_i}{\partial y_{i-1}} &= -\dfrac{\dfrac{\partial \varphi_i}{\partial y_{i-1}} q_{iy} - \dfrac{\partial \phi_i}{\partial y_{i-1}} q_{ix}}{f_i(x_i, y_i) q_{iy} - g_i(x_i, y_i) q_{ix}} \end{cases}, \tag{3.49}$$

$\partial y_i/\partial x_i = \partial y_i/\partial x|_{q_i(x_i,y_i)=0 \text{ and } x=x_i}$; q_{ix} and q_{iy} are the components of a vector tangent to the switching line.
Then $\partial y_1/\partial x_0 = \partial y_1/\partial x_1 . \partial x_1/\partial x_0$.

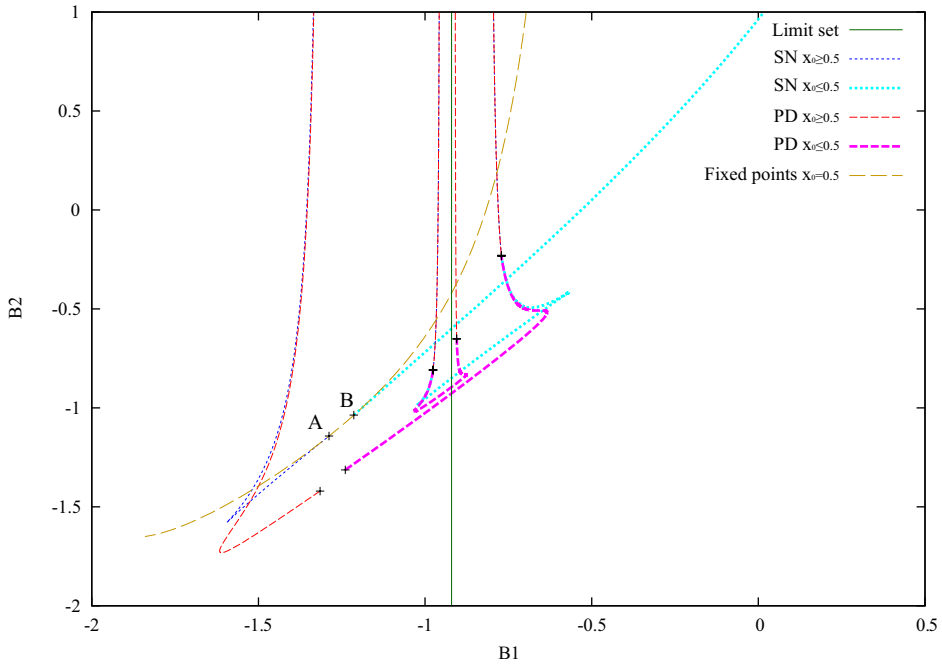


Figure 3.22: Bifurcation diagram in the B_1/B_2 parameter plan

We numerically integrate the required elements:

$$\begin{aligned}
&\frac{d}{dt}\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f_i(x,y) \\ g_i(x,y) \end{bmatrix} \quad \begin{matrix} \text{State 1: from } x_0 \\ \text{State 2: from } x_1 \end{matrix} \\
&\frac{d}{dt}\begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix} = \frac{\partial f_i(X)}{\partial X}\begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix} \\
&\text{where } \begin{bmatrix} \dfrac{\partial\varphi_i}{\partial x_{i-1}} & \dfrac{\partial\varphi_i}{\partial y_{i-1}} \\ \dfrac{\partial\phi_i}{\partial x_{i-1}} & \dfrac{\partial\phi_i}{\partial y_{i-1}} \end{bmatrix}_{t=0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned} \tag{3.50}$$

Along the bifurcation curves, not only B_1 and B_2, but also $x_0 = x_2$, hence x_1 and y_1 vary as well. As long as $x_1 \geq 0.5$, we treat the system just like the standard Alpazur Oscillator. When $x_1 \leq 0.5$, we take the change of switching condition into account and treat it with the equations previously introduced. Remains the case of all the orbits that go through exactly $x_1 = 0.5$ which corresponds to the non-smooth point of the the switching threshold and requires a special treatment. For all these orbits, we can take both the left and right side of the first switching limit, resulting in two eigenvalues. There are many curves in the $\{B_1, B_2, x_0\}$ space that describe such fixed points, so we chose to display just one in our bifurcation diagram (see Fig. 3.22.) Each point having two eigenvalues, we can find two different points along this curve corresponding to a

saddle-node bifurcation (example with points A and B in Fig. 3.23). The first one will correspond to the end of the SN bifurcation curve of orbits with $x_1 \geq 0.5$, and the other one will correspond to the beginning of the SN bifurcation curve with $x_1 \leq 0.5$. In between, there exist fixed points, but there is no parameter values corresponding to a SN bifurcation.

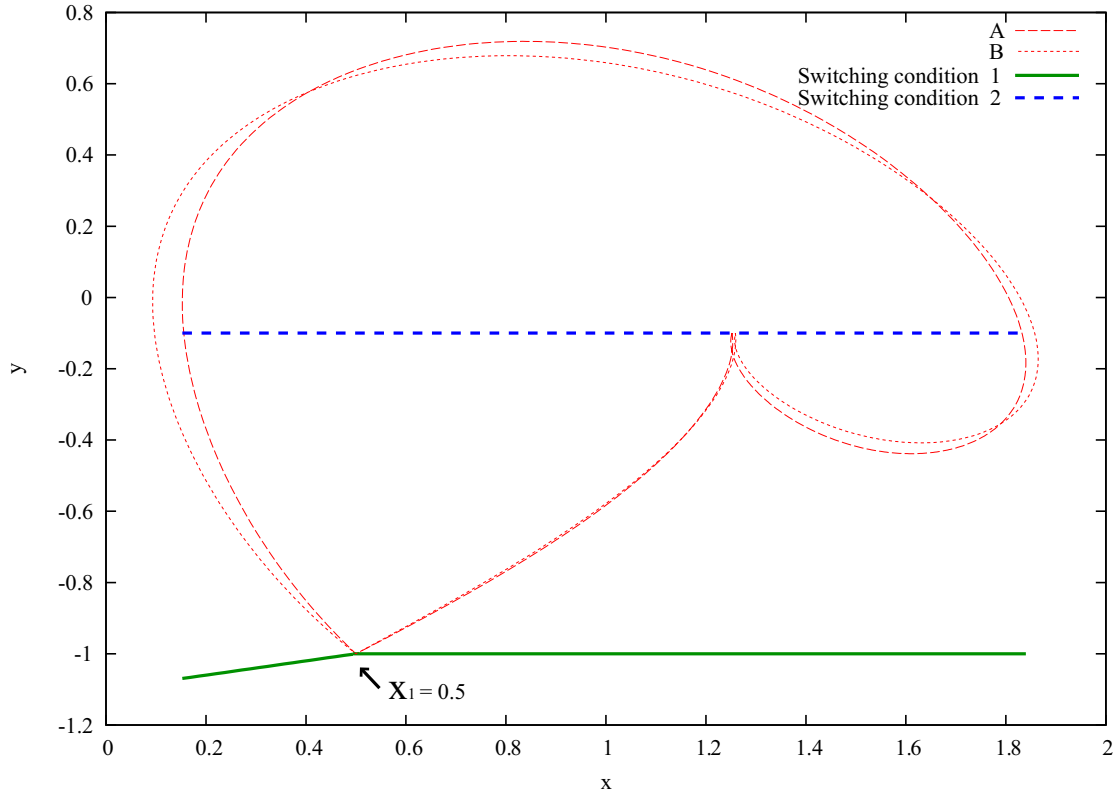


Figure 3.23: A and B points (from Fig. 3.22) at $x_1 = 0.5$

In order to compute the precise parameter values corresponding to those points, we used the Poincaré map to our advantage: instead of starting from X_0, we started from $X_1 = \{0.5, -1.0\}$, and then computed the parameter values corresponding to a SN bifurcation, once using $q_1(x, y) = y + 1.1 - 0.2x$ and once using $q_1(x, y) = y + 1.0$.

Note that in Fig. 3.22, some points are so close to each other that the distinction is not visible at this scale. In this system, the closer we get to the limit set at $B_1 \approx -0.92$, the stronger the influence of B_1 becomes, making the influence of the switching limit on the bifurcation line appear relatively weak. Yet, the discontinuity exists.

One more note, the degree of non-smoothness, or even the gap (in case of a discontinuity) in the switching threshold obviously has a strong impact on the resulting discontinuity of the bifurcation diagram. We can easily imagine that some collision bifurcation might occur when reaching such point, making the bifurcation line stop.

3.4 3D Alpazur oscillator

The Alpazur oscillator evolves in a hybrid space, with two continuous variables which we named so far x and y, and one discrete variable represented by what we called state 1 and 2 (referred as i). Though this is enough to obtain chaos, we will now increase the dimension of the continuous space in order to show how to apply the Poincaré sections and handle the resulting 2D map. This should be enough to prove the generality of the method and illustrate higher dimensional systems analysis.

3.4.1 Model description

In order to extend the Alpazur oscillator into a 3-dimensional system, we add an extra capacitor C_2 (as shown Fig. 3.24.)

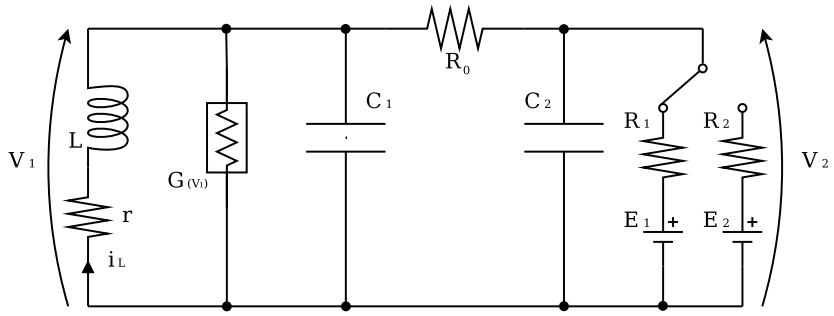


Figure 3.24: Electronic implementation of the 3D Alpazur oscillator.

This makes it look closer to a Chua circuit, only the nonlinear component G is on the side of the capacitor C_1 rather than C_2.

We have now 3 continuous variables: i, v_1, and v_2; while the discrete variable is still the state i ($\in \{1; 2\}$).

$$\begin{cases} L\dfrac{di}{d_t} = -ri - v_1 \\ C_1\dfrac{dv_1}{d_t} = i - g(v_1) + \dfrac{v_2 - v_1}{R_0} \\ C_2\dfrac{dv_2}{d_t} = \dfrac{v_1 - v_2}{R_0} + \dfrac{E_i - v_2}{R_i} \end{cases} \tag{3.51}$$

where the nonlinear resistor characteristic is:

$$g(v_1) = -a_1 v_1 + a_3 v_1^3, \quad \text{with} \quad a_1, a_3 > 0. \tag{3.52}$$

The variables are normalized:

$$x = i\sqrt{L} \qquad y = v_1\sqrt{C_1} \qquad z = v_2\sqrt{C_2}$$

$$r = r\sqrt{\frac{C_1}{L}} \qquad \tau = \frac{t}{\sqrt{LC_1}} \qquad \alpha = \frac{C_2}{C_1}$$

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad b = \left(a_1 - \tfrac{1}{R_0}\right)\sqrt{\frac{L}{C_1}} \quad c = 3\frac{a_3}{C_1}\sqrt{\frac{L}{C_1}} \quad d = \frac{1}{R_0}\sqrt{\frac{L}{C_1}}$$

$$A_i = \left(\tfrac{1}{R_0} + \tfrac{1}{R_i}\right)\sqrt{\frac{L}{C_1}} \quad B_i = \sqrt{L}\frac{E_i}{R_i}$$

Which results in the following set:

$$\begin{cases} \dfrac{dx}{dt} = -rx - y \\ \dfrac{dy}{dt} = x + by - \dfrac{c}{3}y^3 + \dfrac{d}{\sqrt{\alpha}}z \\ \dfrac{dz}{dt} = \dfrac{1}{\sqrt{\alpha}}\left(dy - \frac{A_i}{\sqrt{\alpha}}z + B_i\right). \end{cases}$$

Finally, let us fix a few parameters:

$$\begin{aligned} a_1\sqrt{\frac{L}{C_1}} = 1 \quad &\Longrightarrow b = (1-d) \\ c = 1 \qquad &A_i = \frac{1}{R_i}\sqrt{\frac{L}{C_1}} + d \\ d = \frac{1}{10} \qquad &\alpha = \frac{1}{100}. \end{aligned}$$

Then we obtain:

$$\begin{cases} \dfrac{dx}{dt} = f_i(x,y,z) = -rx - y \\ \dfrac{dy}{dt} = g_i(x,y,z) = x + \dfrac{9}{10}y - \dfrac{1}{3}y^3 + z \\ \dfrac{dz}{dt} = h_i(x,y,z) = y - 100A_iz + 10B_i. \end{cases} \tag{3.53}$$

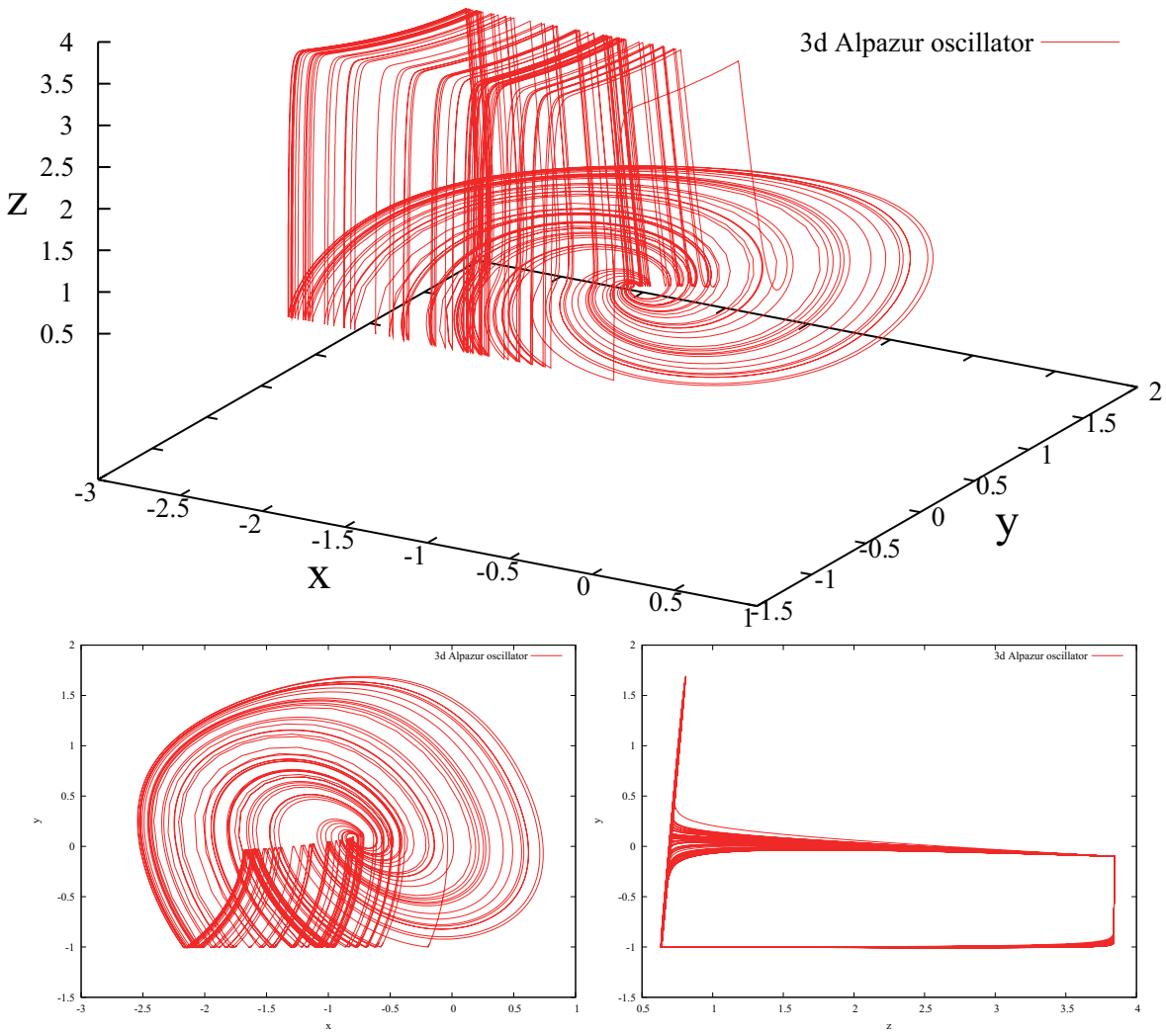
By choosing the capacity C_2 to a value small enough in comparison with C_1, we can expect the dynamics involving v_2 to be much faster than those of v_1. This way, we keep the overall behavior of the system close to the original Alpazur oscillator.

The switching conditions are kept unchanged:

$$q_i(X) = y - h_i, \qquad \text{with} \qquad h_1 = -1, \quad \text{and} \quad h_2 = -0.1. \tag{3.54}$$

And given the right parameter values, the system exhibits chaotic orbits, as expected (see Fig. 3.25):

$r = 0.1 \quad A_1 = 0.15 \quad A_2 = 1.3 \quad B_1 = 1.044 \quad B_2 = 50$



(a) In x/y it is very close to the original Alpazur (b) In z/y we can clearly see the hysteresis

Figure 3.25: Sample of chaotic orbit of the 3D Alpazur oscillator.

3.4.2 Analysis formulation

We naturally consider the Poincaré map as follows:

$$\begin{aligned} T_1: \quad \Pi_0 &\to \Pi_1 \\ X_0 &\mapsto X_1 \\ T_2: \quad \Pi_1 &\to \Pi_0 \\ X_1 &\mapsto X_2 \\ T = T_2 \circ T_1: \quad \Pi_0 &\to \Pi_0 \\ X_0 &\mapsto X_2 \end{aligned} \tag{3.55}$$

Considering the switching conditions, we define the projection p:

$$\begin{array}{rccl} p: & \Pi_0 & \rightarrow & \Sigma_0 \\ & X & \mapsto & U = \begin{bmatrix} x \\ z \end{bmatrix} \end{array} \tag{3.56}$$

3.4.3 Fixed points

Beside the extra dimension (having to deal with matrices instead of real numbers), the approach is still the same: computing dU_2/dU_0.

$$\frac{\partial U_2}{\partial U_0} = \frac{\partial U_2}{\partial U_1}\frac{\partial U_1}{\partial U_0}. \tag{3.57}$$

For each State i we compute:

$$\begin{aligned} \frac{dU_i}{dU_{i-1}} &= \frac{\partial \varphi_i}{\partial U_{i-1}} + \begin{bmatrix} f_i(x_i, y_i, z_i) \\ h_i(x_i, y_i, z_i) \end{bmatrix} \frac{\partial \tau_i}{\partial U_{i-1}} \\ \frac{\partial \tau_i}{\partial U_{i-1}} &= \frac{-\dfrac{\partial \phi_i}{\partial U_{i-1}}}{g_i(x_i, y_i, z_i)}. \end{aligned} \tag{3.58}$$

Note here that φ is 2-dimensional (for x and z).

We numerically integrate the required elements as follows:

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} &= \begin{bmatrix} f_i(x,y,z) \\ g_i(x,y,z) \\ h_i(x,y,z) \end{bmatrix} \quad \begin{array}{l} \text{State 1: from } \begin{bmatrix} x_0 \\ b \\ z_0 \end{bmatrix} \\ \text{State 2: from } \begin{bmatrix} x_1 \\ h \\ z_1 \end{bmatrix} \end{array} \\ \frac{d}{dt}\begin{bmatrix} \dfrac{\partial \varphi_i}{\partial U_{i-1}} \\ \dfrac{\partial \phi_i}{\partial U_{i-1}} \end{bmatrix} &= \begin{bmatrix} \dfrac{\partial f_i}{\partial x} & \dfrac{\partial f_i}{\partial y} & \dfrac{\partial f_i}{\partial z} \\ \dfrac{\partial h_i}{\partial x} & \dfrac{\partial h_i}{\partial y} & \dfrac{\partial h_i}{\partial z} \\ \dfrac{\partial g_i}{\partial x} & \dfrac{\partial g_i}{\partial y} & \dfrac{\partial g_i}{\partial z} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial U_{i-1}} \\ \dfrac{\partial \phi_i}{\partial U_{i-1}} \end{bmatrix} \\ \text{where } \begin{bmatrix} \dfrac{\partial \varphi_i}{\partial U_{i-1}} \\ \dfrac{\partial \phi_i}{\partial U_{i-1}} \end{bmatrix}_{t=0} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \end{aligned} \tag{3.59}$$

We now use the Newton method to compute the correction to be applied:

$$U_0' = U_0 - \frac{U_2 - U_0}{\dfrac{\partial U_2}{\partial U_0} - 1}. \tag{3.60}$$

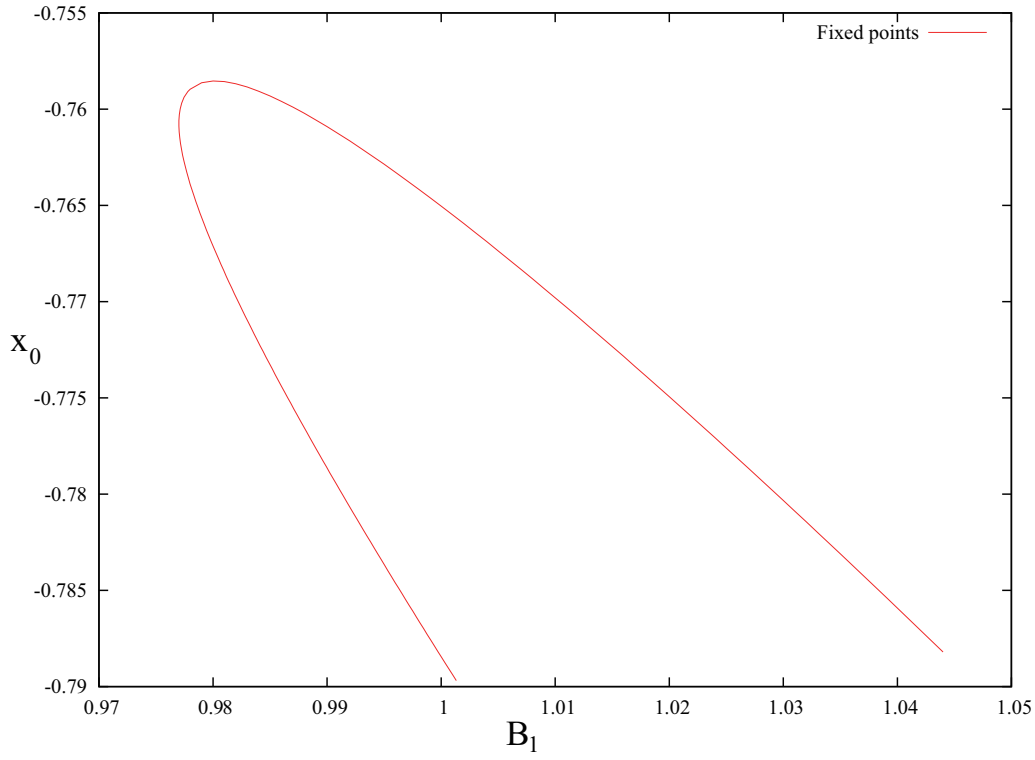


Figure 3.26: Fixed points for the 3d Alpazur oscillator ($B_2 = 50$).

We obtain a diagram of fixed points (such as in Fig. 3.26.)

Though incomplete, this portion of fixed points curve clearly reminds us of the kind we found in the original Alpazur oscillator. The kind of fixed point orbit calculated Fig. 3.27 also shows signs of interactions between equilibrium point at state 1 and the second switching limit. It would be a possible work in the future to investigate the influence of C_2 over the bifurcation structure, particularly on the cusps cascade.

3.4.4 Bifurcation points

This time, DT_l is a matrix, so we actually need to compute the determinant for $\chi(\mu)$:

$$\chi_l(\mu) = \det\left(\frac{\partial U_m}{\partial U_0} - \mu I_2\right) = 0. \tag{3.61}$$

As before, monitoring the eigenvalues reveals candidate bifurcation points.

By proceeding exactly as in the previous systems, we compute the Jacobian matrix:

$$\begin{bmatrix} \frac{dx_2}{dx_0} & \frac{dx_2}{dz_0} & \frac{dx_2}{dB_1} \\ \frac{dz_2}{dx_0} & \frac{dz_2}{dz_0} & \frac{dz_2}{dB_1} \\ \frac{d\chi}{dx_0} & \frac{d\chi}{dz_0} & \frac{d\chi}{dB_1} \end{bmatrix} \quad \text{with} \quad \mu = 1. \tag{3.62}$$

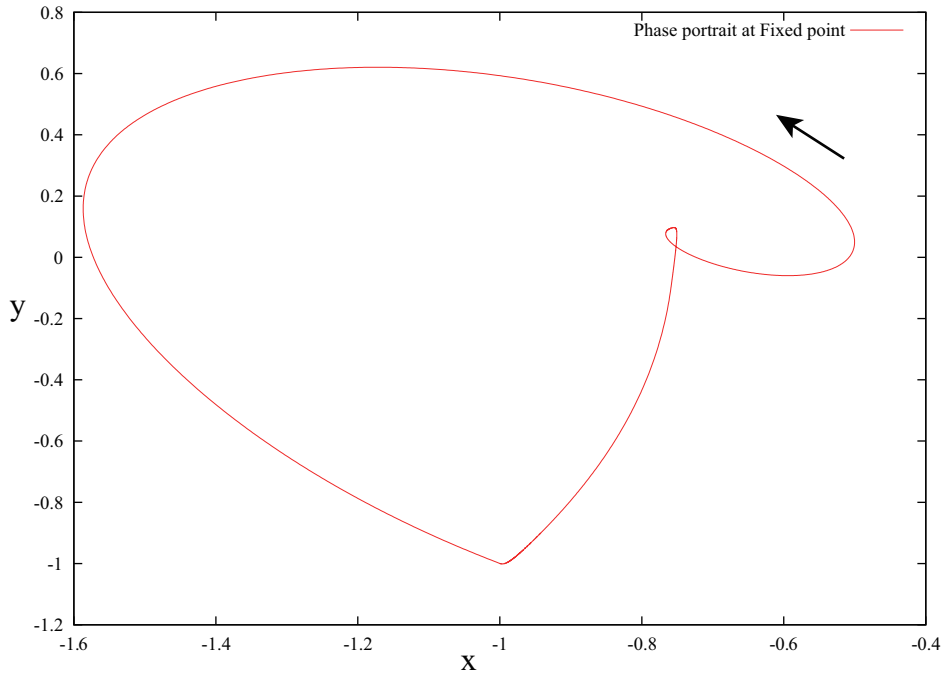


Figure 3.27: The influence of the unstable equilibrium point is visible at the initial point.

We converged to the following point and parameter value:

$x_0 = -7.6071586869e-01$
$z_0 = 3.8452087045e+00$
$B_1 = 9.7701187477e-01$

Which is the exact extreme B_1 value of the fixed points curve (see Fig 3.26.)

This demonstrates the efficiency and flexibility of our method in terms of n-dimensional systems. The complete analysis of this system may be the topic of a separate publication, and the results gathering should be straight forward from this point on.

3.5 Recap

In this chapter, we have put the analysis method to use and showed how it can be adapted to apply to a variety of scenarios. We naturally put the accent on the switching conditions which are the very essence of hybrid systems: increased the number of discrete states, used linear and nonlinear switching thresholds, and even unsmooth ones. The latter case is very close to discontinuous cases which can be handled in a similar way. Eventually, the 3-dimensional version of the system shows the method can be scaled to n-dimensional systems easily.

Overall, these systems are very similar in their structure and behavior, so they naturally appear to have comparable bifurcation structures. We have briefly addressed the question of the presence of a limit set with a cascade of cusps in Sec. 3.2, which was found in all versions of the Alpazur oscillator. There are too few results for the 3D version to confirm such structure, but the study of the influence of the extra capacity over the limit set could be an interesting opportunity for future work.

This framework is simple, yet proved efficient, flexible, and easy to implement as part of a computer program, which has been so far our major concerns.

Chapter 4

Detail of relevant numerical methods

This section is dedicated to the quick review of a few useful numerical tools and approaches we used to obtain our results. It is not meant to be exhaustive and might not be optimal in all considerations, but it is there to provide a set of accessible methods to handle some of the key steps of the numerical bifurcation analysis we consider.

The first one is a numerical differentiation, which may appear as a quite naive approach, but turned out so efficient at addressing one of the main drawbacks of our method (second derivative elements) that we decided to try and integrate it.

The second numerical method we used is the well known variable step-size Runge Kutta. We review it and introduce a slight modification in an attempt to improve the precision and gain some numerical error control.

The last section is simply a few considerations concerning tracing algorithm when solving diagrams.

4.1 Numerical differentiation for derivation approximation

Obtaining an analytical expression for the second row of the matrix (14) like in the Eq. (8) is possible, and this is how Kousaka *et al* [1999] obtained his results.

However, we can see in Eq. 8 that some elements such as $\partial\tau_i/\partial X_{i-1}$ are heavily depending on the switching conditions, making it difficult to express in a general form. In order to obtain the Jacobian matrix (14), such terms must be derived once more.

The complexity of the resulting equation depends mainly on the complexity of the switching condition $q_i(X)$. Even when expressing the Jacobian matrix of the 2 state Alpazur Oscillator (very simple switching conditions), one must spend a fair amount of time and paper to write all the equations down. The complexity of $\partial\tau_i/\partial X_{i-1}$ in Eq. (8) is already impossible to generalize without fixing some restrictions on the type of possible switching conditions, while its second derivatives such as $\partial^2\tau_i/\partial X_{i-1}^2$ or $\partial^2\tau_i/(\partial X_{i-1}\partial\lambda)$ require double the effort, most likely presenting a lot of case by case issues.

By making use of numerical integration, we are certain that a computer algorithm will be involved in any usage of this method. Hence we want to make it:

- easy to implement and generalize;
- easy to use;
- fitted to computer-based tools in an efficient way.

To generalize all the possible switching scenarios and then implement them is complicated; and asking the user to input his own equations for the Jacobian matrix is far from being user friendly. Also, such sequential calculations are no longer fitted to recent hardware that tends to handle parallelized computations better and better.

We propose a simple, yet efficient method which consists in approaching the tangent by differentiation, performing a multiple integration using shifted input variables or parameters.

We compute the following elements the same way we would in the fixed point algorithm:

$$\begin{array}{lll} U_m(U_0,\lambda), & U_m(U_0+\Delta U,\lambda), & U_m(U_0,\lambda+\Delta\lambda), \\ DT_l(U_0,\lambda), & DT_l(U_0+\Delta U,\lambda), & DT_l(U_0,\lambda+\Delta\lambda). \end{array} \tag{4.1}$$

We derive the Jacobian matrix elements by differentiation:

$$\begin{aligned}
\frac{\partial U_m}{\partial U_0} &\approx \frac{U_m(U_0 + \Delta U, \lambda) - U_m(U_0, \lambda)}{\Delta U} \\
\frac{\partial U_m}{\partial \lambda} &\approx \frac{U_m(U_0, \lambda + \Delta\lambda) - U_m(U_0, \lambda)}{\Delta\lambda} \\
\frac{\partial DT_l}{\partial U_0} &\approx \frac{DT_l(U_0 + \Delta U, \lambda) - DT_l(U_0, \lambda)}{\Delta U} \\
\frac{\partial DT_l}{\partial \lambda} &\approx \frac{DT_l(U_0, \lambda + \Delta\lambda) - DT_l(U_0, \lambda)}{\Delta\lambda}.
\end{aligned} \tag{4.2}$$

Such approach introduces some questions, such as how to determine a relevant ΔU or $\Delta\lambda$. Though it is not a truly refined approach, we use at this point a fraction of the correction of U and λ as ΔU and $\Delta\lambda$ for the next iteration. Based on the assumption that we do converge to the solution, the next correction is expected to be smaller than the previous one, hence a smaller differentiation step: the closer ΔX and $\Delta\lambda$ gets to the desired correction, the faster we converge.

The next method to determine ΔU and $\Delta\lambda$ that we consider will take into account the precision achieved numerically and the precision expected by the user to choose relevant values. This implies error control concepts not in the scope of this paper, and the results associated are yet to be gathered, so this topic will be addressed in another publication.

Such a method is simple and adapted to numerical algorithms. It is easily parallelized: one process can be assigned to the computation of one element, and U_0 being a vector, $\partial DT_l/\partial U_0$ means n elements. Those n elements would need to be integrated anyways if we were using the analytical approach. This means that, from a computation amount perspective, we are not wasting as much computation time as one might think when considering multiple integrations. During our analysis, we used a multithreaded program which was able to exploit all 4 cores of the machine on which it was running, yielding more than satisfying performance.

Method	Elements to integrate	Elements count
Numerical	$X,$ $\dfrac{\partial\varphi}{\partial x_0}\cdots\dfrac{\partial\varphi}{\partial x_{n-1}}$	$n+1$
Analytical	$X,$ $\dfrac{\partial\varphi}{\partial x_0}\cdots\dfrac{\partial\varphi}{\partial x_{n-1}},$ $\dfrac{\partial^2\varphi}{\partial x_0^2}\cdots\dfrac{\partial^2\varphi}{\partial x_0\partial x_{n-1}},$ $\vdots$ $\dfrac{\partial^2\varphi}{\partial x_{n-1}^2}$	$\dfrac{n(n+5)}{2}+2$

The expression of $\partial^2 X_i/\partial X_{i-1}^2$, from which $\partial DT_l/\partial U_0$ is derived, is non trivial in its analytical form. Also, the numerical method is a trade-off: reducing the complexity by increasing the amount of computations. Fewer and simpler elements are integrated multiple $(n+2)$ times with shifted initial values or parameters. This becomes an asset if we parallelize these integrations, returning results faster for two or more threads.

4.2 Variable step Runge-Kutta based method

The Runge-Kutta method is well known for its precision and efficiency. For our analysis tools, we need both precision, and adaptability to various systems. In order to dynamically adapt the precision / computation time ratio, we will need an adaptive-step approach. Fig.4.1 illustrates this point.

4.2.1 Standard variable step Runge-Kutta method

Some of the main benefits of a variable step integration method are error estimation and control, and increase in processing speed.

The idea is to regularly evaluate the integration error in order to optimize the step size based on two constraints: we want the step to be as large as possible to reduce the necessary CPU time, we want this step size to be small enough to preserve a certain precision level.

In other words, in regions of near-linearity, the step size can increase for better performance without sacrificing too much precision. In regions of strong nonlinearity, we will reduce the step size to reduce the integration error.

The following approach is based on a Runge-Kutta 4 (RK4) method. We compute the next value twice:

- once in a single step of size h: $X^{(1)}(t+h) = \mathrm{RK4}\left(X(t), h\right)$
- once in two steps of size $h/2$: $X^{(2)}(t+h) = \mathrm{RK4}\left(\ \mathrm{RK4}\left(X(t), h/2\right)\ , h/2\right)$

as illustrated in Fig. 4.2.

When considering the error resulting from a RK4 integration step, we figure that the error expression is of the form: $\varepsilon = ch^5$, where c is nearly constant.

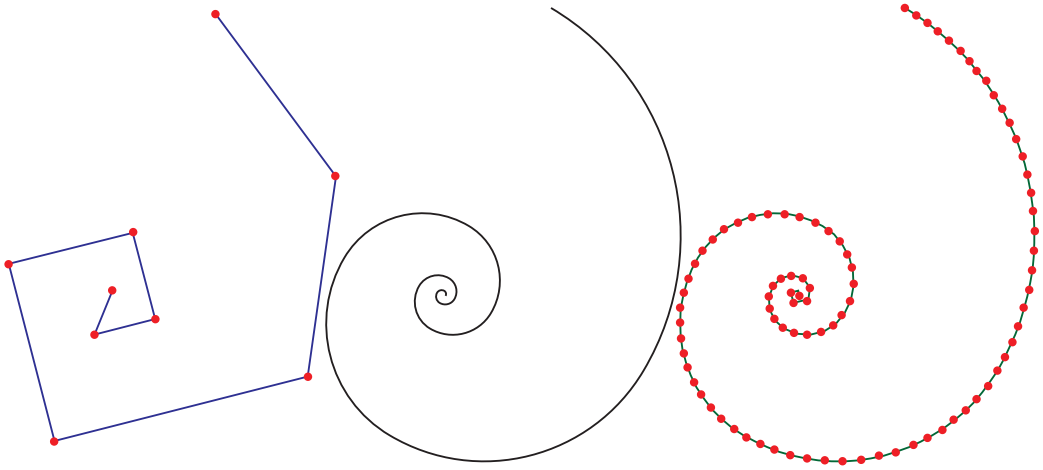
We establish a target error ε_t and call the error estimation ε_e.

$$\begin{gathered} \varepsilon_e = \|X^{(2)} - X^{(1)}\| \\ \varepsilon_e = c(h)^5 \quad \text{and} \quad \varepsilon_t = c(h^*)^5 \quad \Rightarrow \quad h^* = h\left(\frac{\varepsilon_t}{\varepsilon_e}\right)^{1/5}, \end{gathered} \tag{4.3}$$

Where h^* is a step size closer to the optimal value.

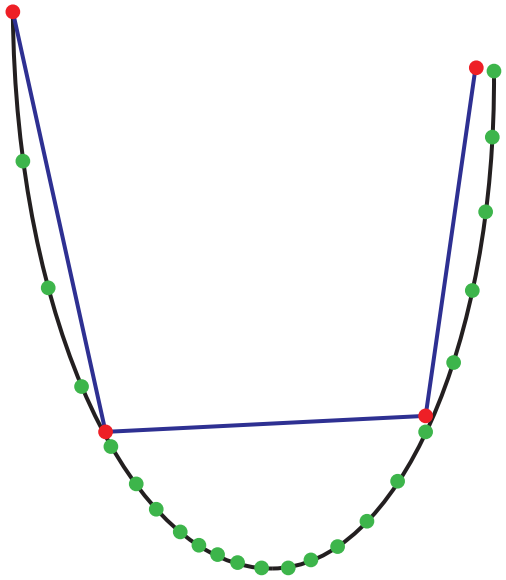
Finally, in order to increase the tolerance, an arbitrary coefficient is introduced:

$$h^* = 0.9 \times h\left(\frac{\varepsilon_t}{\varepsilon_e}\right)^{1/5} \tag{4.4}$$



(a) A step-size too big will introduce numerical error.

(b) A step-size too small will waste CPU time by performing an unnecessary amount of computation, hence negatively affect the performance. In extreme cases, it also introduces and accumulates unnecessary numerical error.



(c) A proper variable step-size approach will adapt to the system to obtain both acceptable precision and good performance.

Figure 4.1: A few scenarios to show how precision needs may vary.

4.2.2 n-iteration windowed RK4

For simple simulations, the previous method usually offers satisfying performance, while keeping the error under tolerable levels.

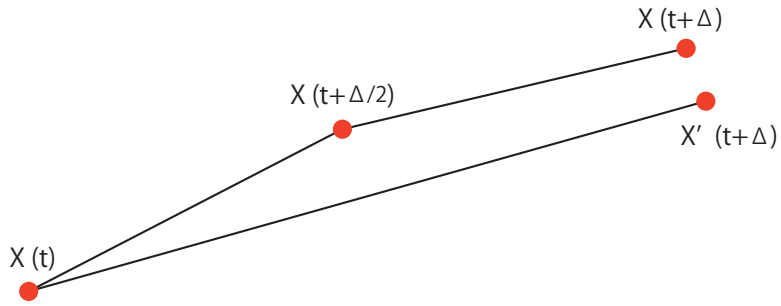


Figure 4.2: Computation of the same point with two different precisions in order to obtain an approximation of the error.

On the other hand, a bifurcation analysis requires a more significant degree of precision. In many highly non-linear systems, this becomes a challenge due to numerical error and related phenomena (machine epsilon, error propagation and others).

More specifically, in the standard variable step Runge-Kutta method, the operation where we divide by $\|X^{(2)} - X^{(1)}\|$ often leads to division by zero issues.

To ensure a more significant difference, we propose using a windowed approach, with the computation of the n next points instead of just one, as shown in Fig.4.3. The drawback is, if the result fails to comply with the desired precision requirements, the whole window of points has to be recomputed with higher precision, instead of one single point.

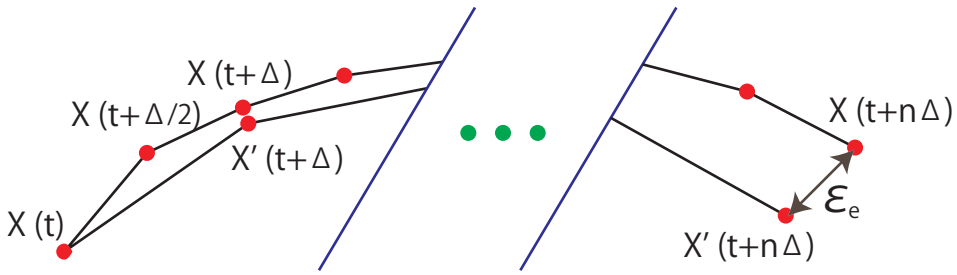


Figure 4.3: Computation of n steps with 2 different precisions to evaluate the error over the whole curve segment.

The number of iterations n can be fixed, or dynamically adjusted using a failure feedback control approach.

The way h is adjusted is basically the same:

$$\begin{aligned} \varepsilon_e &= \|X^{(2)}(t+nh) - X^{(1)}(t+nh)\| \\ h^* &= 0.9 \times h \left(\tfrac{\varepsilon_t}{\varepsilon_e}\right)^{1/5} \end{aligned} \tag{4.5}$$

This approach preserves the adaptability of the original method, does not sacrifice much of the performance in terms of computation time, while exploiting the machine precision to its limits.

4.3 Linear prediction and tracing algorithm

There exist many predication algorithms in order to compute and trace effectively an initially unknown curve.

How such computations are handled are most important as it will affect the result in the following:

- performance: computing too many points in a nearly linear region is a waste of CPU time;
- precision: computing too few points in a parameter-sensitive region can lead to insufficiently precise curves;
- stability: our computations are based on a Newton method, so if the predicted point is actually too far from the solution, the computation may end up diverging.

The two first points can be handled using a variable step-size, just like in section 4.2.

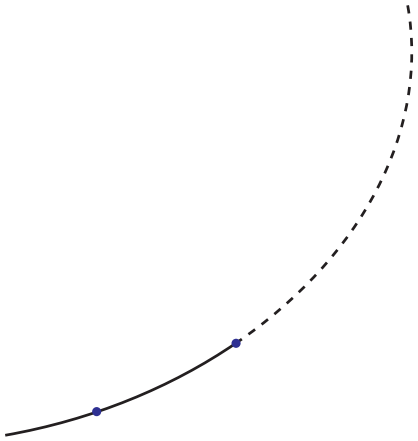
The prediction approaches we used are purely linear, though depending on the available data some more complex approaches can be used.

Next is how we may handle a computation failure. A divergence is most of the time due to a prediction with an error too big. This is the case when the step-size is too large for the Newton method to correct such variations. Even when the precision appears satisfactory for the user, it is not relevant to the parameter sensitivity of the system.

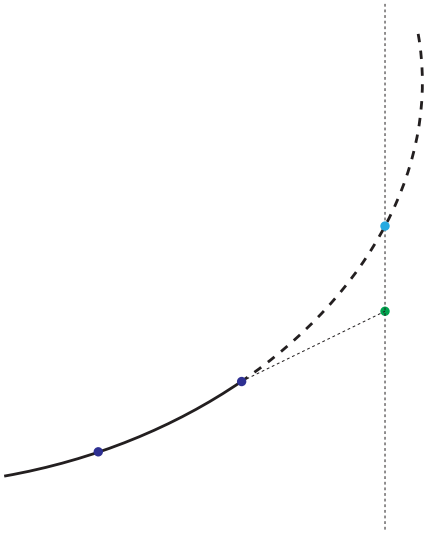
4.3.1 Prediction approach

For extremely precision-dependent and/or time-consuming computations, there exists a variety of prediction methods based on the history of results. In our specific case, we use a Newton method to refine the predicted values, which means that we may not need a very high level of prediction precision, just enough to be in the convergence domain. Also, the computation of the curve being an iterative process, we want to pick a step size small enough to have results as close as possible to a smooth curve and not a broken line. This condition is enough to assume the curve is locally near-linear. Given these two considerations, we opted for a simple linear projection based on the last two points computed and the current step size.

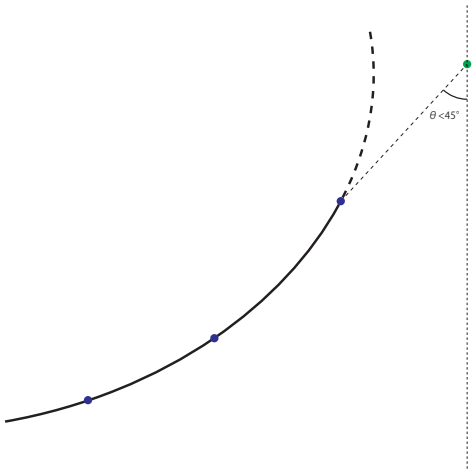
Another aspect of the prediction process is determining, in the variables and parameters space, which values to fix and which ones to refine. When the derivative of the fixed component of the curve changes sign, it can happen that the projection incorrectly predicts a value in a subspace where there is no solution. In order to avoid such situation, we monitor the evolution of the curve and refine the values that vary rapidly (the most likely to be included in the projected subspace).



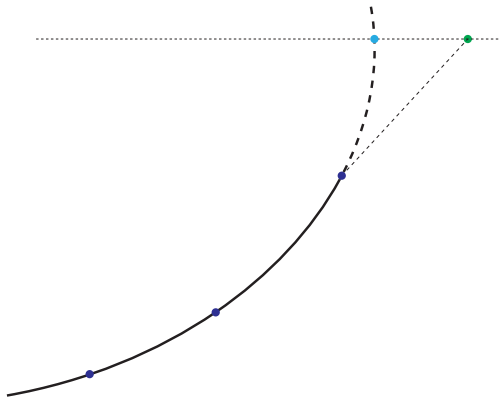
4.4.1: Computation reached the point n.



4.4.2: using a linear projection, we determine an approximation which is refined in the subspace at parameter λ fixed.



4.4.3: The next step will fail if the same parameter λ is picked. By observing the angle θ we have to switch to the other parameter of the space: λ'



4.4.4: Refining the result is successful in the appropriate subspace.

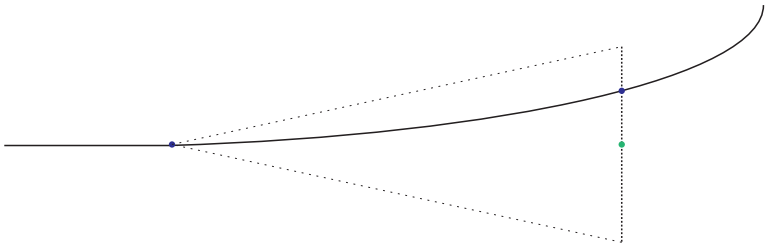
Figure 4.4: Dynamical selection of fixed parameters for computation of the solution.

4.3.2 Step size control

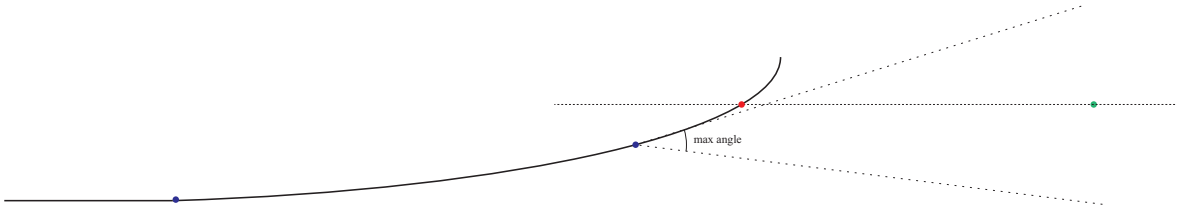
The basic idea is still the same as what is exposed in 4.2.1. We want to minimize the error in all point of the curve while saving computation time by increasing the step size in nearly linear regions. For this purpose, we defined the level of nonlinearity as

the angle between two segments of the curve (actually a broken line). By fixing a maximum angle and a maximum step size, we set an error tolerance. We also pick a second angle value, the target one, smaller than the maximum, so that we have enough margin to minimize the amount of failures due to unexpected error variations.

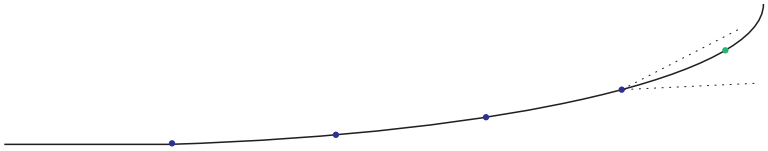
Even if such approach greatly increases the stability of this method, there are still cases of failure to converge, particularly in cases where the curve nonlinearity increases rapidly. With such scenario in mind, we have implemented one last simple mechanism. In most cases, a failure to converge is either directly related to the characteristics of the system (like the presence of a global bifurcation, particularly collisions in hybrid systems), or simply a step size too big. In either case, we usually can gain in precision by simply doing a rollback and trying to recompute the one or two last points. This way, the algorithm has multiple attempts to handle a point presenting a computation challenge, and will be allowed to give up only when reaching a specific limit of time or precision as desired by the user.



4.5.1: Upon reaching a highly nonlinear region, the step size is too large, yet the next point may fall into an acceptable angle.



4.5.2: However, using the projection mentioned in 4.3.1, whatever the current step size, we cannot find a point within the set angular tolerance.



4.5.3: Instead, we have to rollback to the last computation, and reprocess it with a smaller step size.

Figure 4.5: Illustration on how a rollback strategy helps recover from some prediction errors.

Chapter 5

Conclusion

Using an approach that is completely adapted to computer, we propose a method to study the bifurcations of virtually any hybrid system. Given that this method can be automatized to a high degree, this promises to unload most of the analysis work from the user to the computer. We have covered with a particular attention how to handle performance and precision issues, and illustrated the whole process with results of a representative set of various kinds of switching cases in a piecewise-smooth system: the Alpazur oscillator.

There are many hybrid systems still undergoing bifurcation analysis in the field of mechanics or electronics and many more waiting to be considered. This method lifting many constrains, particularly in terms of dimensions and complexity, future work may be to analyze such systems. In other words, we first need to validate the method in a wider range of systems. It implies a work of comparison with real-world results and measurements. This approach is also very generic and could be applied to more than nonlinear systems, in which case a study of efficiency and comparison to other methods could also be of interest. Relevant industrial application related systems include multi-cell chopper like the ones analyzed by Berthoux and Barbot in [22], PWM inverters the like of B. Robert and C. Robert studied collision bifurcations in [23], or mechanical ones as cam-follower rigs with impacts that Alzate, di Bernardo and al. study in [24].

As for the method itself, of course each portion of the algorithm can be improved and handled in a different or even more efficient way, so the method is still candidate to improvements, but beyond optimization, the most obvious step to be taken next would be considering hybrid systems, list all the possible switching scenarios, and find the simplest generic expression. Since this is, as we could see in our examples, one of the most crucial part of the model construction, it would bring the final stone required to build a stand-alone and automatized bifurcation analysis computer tool for hybrid systems. There is also a number of models such as those featuring jumps at switching points, changes of space or discontinuities, and which can definitely be handle using our method. Just like for the various Alpazur extensions, it would be very interesting to detail the model expression and specific attentions that must be considered for such

systems when going through the analysis process.

A few steps, including the first derivation of the differential equations, could be automatized further once the set of switching conditions is clearly established. The tools for the manipulation of symbolic expressions exist, so it really is a work of testing and verification as well.

Finally, the whole group of global bifurcations is left to study and represents a great opportunity to see how this method can be extended further to handle bifurcations such as collisions which characterize hybrid systems, though this might prove much harder to automatize.

Bibliography

[1] di Bernardo, M.; Tse, C. [2002] "Chaos in Power Electronics: An Overview," in *Chaos in Circuits and Systems*, Series B, Vol. 11, pp. 317–340.

[2] Tse, C. [2004] *Complex Behavior of Switching Power Converters* (CRC press) Chap. 3, pp. 57–80.

[3] Banerjee, S.; Chakrabarty, K. [1998] "Nonlinear Modeling and Bifurcations in the Boost Converter," in *IEEE Trans. Power Electronics* Vol. 13(2), pp. 252–260.

[4] Banerjee, S.; Karthik, M.S.; Yuan, G.H.; Yorke, J.A. [2000] "Bifurcations in one-dimensional piecewise smooth maps - theory and applications in switching circuits," in *IEEE Trans. Circuits and Systems I* Vol. 47(3), pp. 389–394.

[5] Acco, P.; Daafouz, J.; Fourniet-Prunaret, D.; Taha, A.K. [2004] "Approche hybride de la stabilité locale de la boucle à verrouillage de phase par impulsions de charge," in *Revue e-STA, ISSN: 1954-3522, http://www.e-sta.see.asso.fr*, Vol. 1(4).

[6] Wiercigroch, M. & de Kraker, B. [2000] *Applied Nonlinear Dynamics and Chaos of Mechanical Systems with Discontinuities* (World Scientific Series on Nonlinear Science, Series A, Vol. 28)

[7] di Bernardo, M.; Budd, C.J.; Champneys, A.R.; Kowalczyk, P. [2008] *Piecewise-smooth Dynamical Systems Theory and Applications* (Applied Mathematical Sciences , Vol. 163)

[8] Kawakami, H. and Lozi, R. [1992] "Switched Dynamical Systems – dynamical of a class of circuits with switch," in *Proc. RIMS Conf. "Structure and Bifurcations of Dynamical Systems"* ed. S. Ushiki (World Scientific), pp.39–58.

[9] Kousaka, T.; Ueta, T.; Kawakami, H. [1999] "Bifurcation of switched nonlinear dynamical systems," in *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing* Vol. 46(7), pp. 878–885.

[10] Kousaka, T.; Ueta, T.; Ma, Y.; Kawakami, H. [2006] "Control of chaos in a piecewise smooth nonlinear system," in *Chaos, Solitons & Fractals* (Elsevier Science) Vol. 27, pp. 1019–1025.

[11] Kabe, T.; Parui, S; Torikai, H.; Banerjee, S; Saito, T. [2007] "Analysis of Piecewise Constant Models of Current Mode Controlled DC-DC Converters," in *IEICE Trans. Fundamentals of Electronics, Communications and Comp. Sci.* E90-A(2), pp. 448–456.

[12] Dankowicz, H. [2007] "On the purposeful coarsening of smooth vector fields," in *Nonlinear Dynamics* (Springer Netherlands), Vol. 50(3), pp. 511–522.

[13] Hiskens, I.A.; Pai, M.A. [2000] "Trajectory sensitivity analysis of hybrid systems," in *IEEE Transactions on Circuits and Systems I*, Vol. 47(2), pp. 204–220.

[14] Carcasses, J.P.; Mira, C. [1991] "An Autonomous Ordinary Differential Equation Generating Alternating Isoordinal Cascades of Cusps in a Bifurcation Plane," in *Proc. Conf. European Conference on Iteration Theory: ECIT'89* (World Scientific) pp. 25–41.

[15] Mira, C.; Taha, A.K. [1991] "Isoordinal Cascades of Cusps with Monotonic Convergence in a Bifurcation Plane, Generated by a Two-Dimensional Diffeomorphism," in *Proc. Conf. European Conference on Iteration Theory: ECIT'89* (World Scientific) pp. 231–238.

[16] Izhikevich, E.M. [2003] "Simple Model of Spiking Neurons," in *IEEE Trans. Neural Networks* Vol. 14(6), pp. 1569–1572.

[17] di Bernardo, M.; Feigin, M. I.; Hogan, S. J.; Homer, M. E. [1999] "Local Analysis of C-Bifurcations in n-Dimensional Piecewise-Smooth Dynamical Systems," in *Chaos, Solitons & Fractals* (Elsevier Science) Vol. 10, No. 11, pp. 1881–1908.

[18] Banerjee, S.; Ranjan, P.; Grebogi, C. [2000] "Bifurcations in two-dimensional maps - theory and applications in switching circuits" in *IEEE Transactions on Circuits and Systems I*, Vol. 47(5), pp. 633–643.

[19] Peterka, F. [1974] "Theoretical analysis of n-multiple (1/n)-impact solutions" in *Acta Tech CSAV*, Vol. 26(2), pp. 462–473.

[20] Sushko, I.; Agliari, A.; Gardini, L. [2006] "Bifurcation structure of parameter plane for a family of unimodal piecewise smooth maps: Border-collision bifurcation curves," in *Chaos, Solitons & Fractals* (Elsevier Science) Vol. 29, pp. 756–770.

[21] Kitajima, H.; Kawakami, H. [1994] "An algorithm tracing out the tangent bifurcation curves and its application to Duffings equation," in *IEICE technical report. Nonlinear problems* Vol. 94(44), pp. 1–7.

[22] Bethoux, O.; Barbot, J.-P. [2002] "Multi-cell chopper direct control law preserving optimal limit cycles," in *IEEE Trans. Control Applications* Vol. 2, pp. 1258–1263.

[23] Robert, B.; Robert, C. [2002] "Border collision bifurcations in a one-dimensional piecewise smooth map for a PWM current-programmed H-bridge inverter," in *Int. J. Control* Vol. 75(16/17), pp. 1356–1367.

[24] Alzate, R.; di Bernardo, M.; Montanaro, U.; Santini, S. [2007] "Experimental and numerical verification of bifurcations and chaos in cam-follower impacting systems," in *Nonlinear Dynamics* (Springer Netherlands), Vol. 50, pp. 409–429

Appendix A

C++ code for Runge-Kutta integration of ODE

The following files implement a standard single step 4th order Runge-Kutta integration, and a variable n-step windowed Runge-Kutta based integration.

The input is an initial point (vector represented by a single column matrix), the description of the system, the function we want to integrate, and a integration step size. For the multiple step integration, a numerical error tolerance and a window size (number of steps to compute) must be input as well.

```
// rungeKutta.hh file
// Description: a collection of functions dedicated to Runge-Kutta
// method-based computations

# ifndef rungeKutta_hh
# define rungeKutta_hh

#include "../my_objs/matrix/mymatrix.hh"
#include "../my_objs/system/system.hh"

/// Standard 4th order Runge-Kutta method
Mymatrix rk4 (const Mymatrix & source, const Mysystem &_system, Mysystem_function
f_i,double dt);

/// Variable step, multiple stepped
Mymatrix *vsRk4 (const Mymatrix & source, const Mysystem &_system, Mysystem_function
f_i,double &dt, double &error, int steps);

# endif
```

```
// rungeKutta.cc file

#include "rungeKutta.hh"

Mymatrix rk4 (const Mymatrix & source, const Mysystem &_system, Mysystem_function
f_i,double dt) {
    Mymatrix xtemp, r1, r2, r3, r4, result;
    r1 = (_system.*f_i)(source);
    xtemp = source + (dt/2.0)*r1;
    r2 = (_system.*f_i)(xtemp);
    xtemp = source + (dt/2.0)*r2;
    r3 = (_system.*f_i)(xtemp);
    xtemp = source + dt*r3;
    r4 = (_system.*f_i)(xtemp);
    result = source + (r1+2.0*(r2+r3)+r4)*(dt/6.0);
    return result;
}
```

```
Mymatrix *vsRk4 (const Mymatrix & source, const Mysystem &_system, Mysystem_function
f_i,double &dt, double &error, int steps) {
    double _error =0;
    int iterations = 0;
    Mymatrix temp_simple, temp_double;
    if (steps¡1) {
      fprintf(stderr,"Attempt to use variable step Runge-Kutta with non strictly-positive
step amount");
      exit(1);
    }
    Mymatrix *results;//(source.height(),steps);
    results = new Mymatrix[steps];
    do {
      temp_simple = source;
      temp_double = source;
      for (int i=0;i¡steps;i++) {
        temp_simple = rk4(temp_simple,_system, f_i,dt);
        temp_double = rk4(rk4(temp_double, _system, f_i,dt/2), _system, f_i,dt/2);
        results[i] = temp_double;
      }
      _error = Mdistance(temp_simple,temp_double);
      if (iterations ++ > DIVERGENCE_NBS) {
        error = _error;
        return results;
      }
      if (fabs(_error)¡fabs(error)/10) {
        dt *= 5.0;
        _error = fabs(error)+1;
      }
      if (fabs(_error)>fabs(error))
        dt = dt/2.0;
      else
        dt *= (0.9*(fabs(error)/fabs(_error)));
    } while (fabs(_error)>fabs(error));
    return results;
}
```

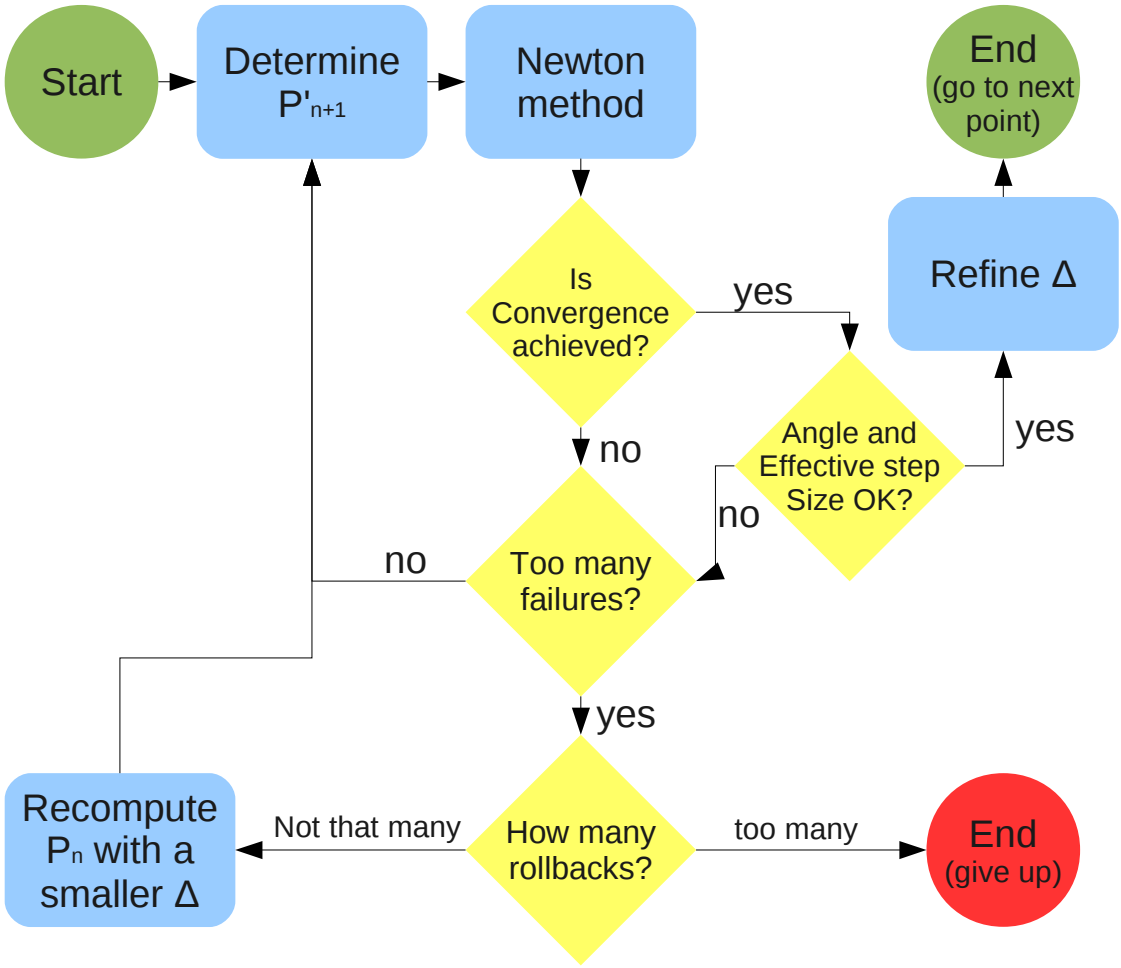
Appendix B

Algorithm for prediction and tracing

This method is based on the assumption that we already know at least two points (vectors of parameter values and initial coordinates P_{n-1} and P_n) of the curve we trace, and have chosen an angle to determine the degree of smoothness we want to achieve, which will control the prediction step size Δ.

1. compute a prediction for the next point: we used a simple linear projection $P'_{n+1} = P_n + \frac{\Delta}{|P_n - P_{n-1}|} * (P_n - P_{n-1})$
2. use the Newton method to obtain a precise P_{n+1}
   - if Newton method fails, set $\Delta = \Delta/2$ and return to 1.
   - if the angle $\widehat{P_{n-1}P_nP_{n+1}}$ is not straight enough, set $\Delta = \Delta/2$ and return to 1.
   - if $\Delta << |P_{n+1} - P_n|$, set $\Delta = \Delta/2$ and return to 1.
3. based on the effective angle, adjust Δ, trying to keep it within reasonable values (not too large nor too small, thought this mostly depends on the system characteristics and the precision achieved)
4. keep track of the number of attempts to rollback to the computation of P_n if we fail too many times, and abort if even such rollback does not help in computing P_{n+1} (within an acceptable time frame: tens of attempts).

Here is an example of flow chart of how to handle this approach:



Appendix C

C++ header of the system object

```
// system.hh file
// Description: describes the system considered for analysis

class Mysystem;

# ifndef mysystem_hh
# define mysystem_hh

#include "../various/result.hh"
#include "../matrix/mymatrix.hh"

// ********************
// Integration specific
// ********************
#define NBSTEPS 50 // size of the computation window for the variable step RK integration
#define TOLERANCE 10e-13 // Error tolerance for a single computation window
#define DIVERGENCE_NBS 10000 // Limit of integration steps before declaring the system non-convergent
#define DIVERGENCE 100 // Limit of any element of the variable vector to assume divergence
#define DT 0.0001 // Initial delta T used for integration, not very important since automatically adjusted

// *************
// Fix+ specific
// *************
#define MAXITER 50 // Limit of iterations for fix and bif before giving up
#define SHIFT_PARAM 10e-6 // Shift in parameter applied for differentiation
```

```
typedef Mymatrix (Mysystem::*Mysystem_function)(const Mymatrix &) const;

class Mysystem {

public:

  ///
  /// public constructor
  Mysystem ();

  ///
  /// setters
  void set_parameters (const Mymatrix &);
  void set_X (const Mymatrix &);
  void set_spp (const int &);
  void set_current_state (const int &);
  void set_nb_states (const int &);
  Mysystem operator= (const Mysystem &source);

  ///
  /// getters
  int nb_states() const;
  int current_state() const;
  int spp() const;
  int sizeX() const;
  Mymatrix X() const;
  Mymatrix parameters() const;

  ///
  /// increment state and return it
  int inc_state();

  ///
  /// Function dX/dt (depends on the current state)
  Mymatrix f_i(const Mymatrix &) const;

  ///
  /// Function d2X/dtdX (depends on the current state)
  /// First column is X, second-on is dX
  Mymatrix df_i(const Mymatrix &) const;
```

```
  ///
  /// Function of the switching condition
  /// Sign determines if a switch is required
  /// Value gives the distance to the switching line
  double switch_cond(const Mymatrix &);

  ///
  /// Function returns the element indice to drop at
  /// a given state
  int rnmo (int state) const;

  ///
  /// Function to determine dXi/dX_i-1 based on the
  /// given state
  Mymatrix dxidximo (const Myresult * result, const Mymatrix * dfi, int step) const;

  ///
  ///
  /// Functions to project from Rn to Rn-1 and back
  Mymatrix PiToSigma (const Mymatrix & X) const;
  Mymatrix SigmaToPi (const Mymatrix & X) const;
  Mymatrix PiToSigma_Jacobian (const Mymatrix & Jacobian) const;

private:

  int _nb_states; // total number of states
  int _current_state; // ...
  int _spp; // nb of switches per period

  Mymatrix _X; // continuous variable (Horizontal vector)

  Mymatrix _parameters; // parameters values (Horizontal vector)

};

# endif
```