Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

QUALITATIVE BIFURCATION DIAGRAMS

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Héctor Rodríguez Rangel

Director de Tesis

Doctor en Ciencias en Ingeniería Eléctrica Juan José Flores Romero

Co-Director de Tesis

Doctor en Ciencias en Ingeniería Eléctrica Claudio Rubén Fuerte Esquivel

Julio 2014

Morelia, Michoacan

A mis padres Héctor y Guillermina,
y a mi hermana Sandra, sin sus
consejos y apoyo incondicional
este paso habría sido posible.

Al Dr. Juan José Flores Romero por ser mi guia
a lo largo de este camino.

Al Dr. Claudio Rubén Fuerte Esquivel por apoyo,
comentarios realizados a este trabajo

Al Dr. Art Farley por sus comentarios y ayuda para
realizar este trabajo durante la estancia en
la Universidad de Oregon.

A mis Amigos, Oscar, Erick y Luis
por su confianza y lealtad

List of Publications

"Qualitative Bifurcation Diagrams"
Rodriguez, Hector; Farley, Arthur M.; Flores, Juan J.; Proskurowski, Andrzej.
Expert Systems Journal. Wiley. ISSN: 1468-0394. August 24th, 2013. DOI: 10.1111/exsy.12043. Impact Factor: 0.769

"Evolutive Design of ARMA and ANN Models for Time Series Forecasting"
Flores, Juan; Graff, Mario; Rodriguez, Hector.
Renewable Energy, Volume 44, August 2012, Pages 225-230, ELSEVIER. ISSN 0960-1481. Impact factor: 2.790.

"Reducing the Search Space in Evolutive Design of ARIMA and ANN Models for Time Series Prediction"
Flores, Juan; Rodriguez, Hector; Graff, Mario.
In Book: MICAI 2010: Advances in Soft Computing. Editors: Grigori Sidorov, Arturo Hernandez Aguirre, Carlos Alberto Reyes Garcia. LNAI 6438. ISBN. 978-3-642-16772-0. Pp 325-336. Springer-Verlag, Berlin, Heidelberg, New York. November, 2010.

"Representación Cualitativa de un Diagrama de Bifurcación a Partir de Datos Cuantitativos"
Rodriguez, Hector; Farley, Arthur M.; Flores, Juan J.
XIII Reunión de Otoño de Potencia, Electrónica y Computación (ROPEC 2011). Morelia, Mexico. November 9-11, 2011.

Resumen

En esta tesis se explora el uso del razonamiento cualitativo para la predicción del comportamiento de un sistema dinámico al variar uno de sus parámetros, basado en una representación cualitativa del diagrama de bifurcación. Tres algoritmos fueron desarrollados para la realización de esta tarea. El primer algoritmo captura las características fundamentales del diagrama de bifurcación y las translada a una representación cualitativa. El segundo algoritmo utiliza la representación cualitativa obtenida del primer algoritmo y realiza la simulación del comportamiento del sistema, dada una secuencia de variaciones en uno de sus parámetros. El tercer algoritmo resuelve el problema opuesto, determina la variación del parámetro partiendo de un estado inicial para llegar a uno final. El primer algoritmo descompone el diagrama en Segmentos Monotónicos (MSs), donde los puntos que conforman a estos segmentos tienen el mismo comportamiento cualitativo (misma naturaleza y comportamiento). Estos MSs son conectados a una red de transiciones. Esta red es utilizada como la principal herramienta de los algoritmos de simulación y planeación. Finalmente se presentan algunos ejemplos que ilustran la representación cualitativa, así como el comportamiento de los algoritmos de simulación y planeación para sistemas dinámicos de un parámetro.

Abstract

This thesis explores the use of qualitative reasoning to predict the behavior of a dynamical system given a change in one of its parameters based upon a qualitative representation of its bifurcation diagram. Three algorithms were developed to perform this task. The first algorithm captures the essential characteristics of the bifurcation diagram in quantitative terms, and translates it into a qualitative representation of the system. The second one uses the qualitative representation obtained from the first algorithm to simulate the behavior of the system given a sequence of parameter adjustments or perturbations. The third algorithm solves the opposite, control problem: it determines what parameters to change to take the system from an initial to a given goal state. The first algorithm segments a quantitative bifurcation diagram into Monotonic Segments (MSs) having the same qualitative behavior (same nature and behavior). These MSs are then interconnected into a transition network. The network is used by the other two algorithms to simulate and plan the behavior of the system from an initial situation. Several examples illustrate the qualitative representations and the behaviors of the simulation and planning algorithms for dynamical systems of one parameter.

Contents

List of Figures

List of Tables

List of Symbols

$AdjacentMS$	Adjacent MS founded
BD	Bifurcation Diagram
$BDPoints$	Bifurcation Diagram Points
BDS	Bifurcation Diagram Segments
$delta$	Parameter used to discriminate points
$DeltaGlue$	Parameter used to approximate segments
$direction$	Defines the slope of a segment
$DownT$	Down Transitions
$EndPoint$	Final segment point
$LeftT$	Left Transitions
$Limits$	defines the boundary of a perturbation
$InitialPoint$	Initial segment point
n_r	Number of parameter landmarks
n_s	Number of segments
n_x	Number of state variables landmarks
MS	Monotonic Segments
MSs	Set of Monotonic Segments
$nature$	Defines the stability of a segment
ODE	Ordinary Differential Equation
$Pert$	Perturbation Transitions
$PertTrans$	Set of perturbation transitions
$PointLeft$	Left end of a MS
$PointRight$	Right end of a MS
Pos	Defines the position of the lowest point
$PosChanges$	Positions where occur a change is slope
QBD	Qualitative Bifurcation Diagram
QRG	Qualitative Representation Generation
QS	Quantitative Space
QS_r	Quantitative Space for the parameter r
QS_x	Quantitative Space for the state variable x
$RightT$	Right Transitions
S	Segment
$Set1$	Set of points observed from below from lowest point to the left
$Set2$	Set of points observed from below from lowest point to the right
$slope$	qualitative slope
$Slopes$	Set of slopes

Chapter 1

Introduction

The goals of science and engineering are to understand, to predict, and to control the behavior of the world. To accomplish these goals, models that represent relevant aspects of real-world phenomena have been developed. One form these models take are what we call dynamical systems, which are used to represent and understand how complex systems change over time. There are different types of dynamical system models: discrete, continuous, linear, and nonlinear. Continuous dynamical systems are often modeled by a set of differential equations [Strogatz94, Boyce01].

Dynamical systems are models that exhibit changes over time. Traditionally, a system's behavior can be derived either by solving differential equations or by simulation. Dynamical system models are formulated in terms of state variables and parameters that may also be functions of time. We often assume that the system's topology does not change. However, this is not true for many complex systems. The mass of a rocket changes as it burns fuel, the characteristics of an electrical machine change as it ages, the load of an electrical power system changes during the day, etc. Changes in such parameters, sometimes even if they are infinitesimal, may cause a dynamical system to exhibit totally different qualitative properties. For instance, a damped mass-spring system may stop oscillating if the damping increases. Changes in the topology of the phase space representation of the dynamical system are known as bifurcations; the values of the parameters for which a bifurcation occurs are called bifurcation points [Strogatz94].

A bifurcation occurs in a dynamical system when its qualitative behavior changes as a result of the variation in one or more of its parameters. The values of the parameters for which the bifurcation occurs is called bifurcation point. A bifurcation diagram is the graphic representation of a dynamical system that plots the set of bifurcation points. The bifurcation points may be obtained after solving algebraically or numerically the differential equations, but sometimes the solution of a dynamical system is complex, inaccurate, or too costly to compute. Some applications have been developed that generate a bifurcation diagram of the dynamical system without solving a differential equation (for example [Flores10, Barrera08] use PSO Particle Swarm Optimization to obtain the bifurcation points).

1.1. Problem Definition

Qualitative Bifurcation Diagrams (QBD) are built by a computer program that takes as input the quantitative data contained in a bifurcation diagram (bifurcation points). It takes the essential features of the system and translates them into a qualitative representation. The quantitative information found in a bifurcation diagram (bifurcation points) does not facilitate the reasoning process needed to determine a behavior of the system as one of the systems' parameter is changed.

Early works [deKleer84, Kuipers94, Forbus84] have used qualitative reasoning to describe the behavior of physical systems. To perform any qualitative reasoning on bifurcation diagrams, we need to take the quantitative data into a qualitative form that is suitable to perform the reasoning tasks of simulation and control planning of the system. In conclusion, the problems we address in this thesis are the following:

1. Given the set of points that compose a quantitative bifurcation diagram, translate it to a language or representation that explicitly represents all qualitative features that make qualitative analysis or reasoning possible. This representation is called a qualitative bifurcation diagram.

2. Given a qualitative representation of a bifurcation diagram and the actions of the system (i.e., perturbations, or increments/decrements in one the system's parameters),

determine the behavior of state variable of the dynamical system (x) as a result of perturbations and changes in systems' parameter (r).

3. Given a qualitative representation of a bifurcation diagram, and a systems' initial and goal states, determine the parameter changes required to take the system from the initial to the goal state.

Throughout the development of this thesis, we have relied on the following assumptions regarding dynamical systems [Flores06]:

- All variables and functions involved in the process are continuous and continuously differentiable.
- The variation of the system's parameters is much slower than the transient time to stabilize the system after a perturbation. According to the definition of landmark (Chapter 3), perturbations always occur at landmarks of r.
- If we need a perturbation to occur in between two landmarks, we simply create another landmark between them and let the perturbation occur at the new landmark.
- Perturbations are small enough so as not to cross other fixed points. If we do not make this assumption, the behavior of the system would depend on the relative magnitude of the perturbation with respect to the distance to the next fixed point in the direction of the perturbation. This last assumption may not hold if a perturbation occurs infinitesimally close to a bifurcation point; we assume perturbation occurs sufficiently far from bifurcation points to prevent this situation from happening.
- Implementation of the system without the perturbation assumption, would lead to branching in the prediction of behavior, i.e. at any perturbation, we would need to consider cases where its magnitude would not reach, exactly meet, or pass each landmark in the direction of the perturbation.

This thesis deals with continuous dynamical systems that can be modeled by ordinary differential equations (ODEs). An ODE is an equation of the form

$$F(t, x, \frac{dx}{dt}, \cdots, \frac{d^n x}{d^n t}) = 0 \tag{1.1}$$

A function $f(t)$ is a solution to Equation (1.1) if it satisfies the original equation.

$$F(t, f(t), \frac{df(t)}{dt}, \cdots, \frac{d^n f(t)}{d^n t}) = 0 \tag{1.2}$$

A phase portrait is a geometric representation of the trajectories that includes all qualitative features that distinguish a dynamical system. Such characteristics include fixed points (attractors and repellers), nullclines, limit cycles, and for more complex (chaotic) systems, even strange attractors (see [Lee93]). Let us consider the nonlinear system represented by $\dot{x} = x^2 - 1$. If we plot x against $\dot{x}$, we get a plot like that of Figure 1.1. Note this is a flow plot, not a phase portrait. In the flow plot of Figure 1.1, trajectories lie on the x axis, pointing to the right when $\dot{x} > 0$, and to the left when $\dot{x} < 0$, as indicated by the arrows. When $\dot{x} = 0$ there is no flow; the places x^* where $\dot{x} = 0$ are called *fixed points*. There are three kinds of fixed points: stable *attractors* (solid black dots), unstable *repellers* (open circles), and semistable (half is open circle and half is solid). As observed in the Figure 1.1 if a perturbation occurs near the black dot (stable or attractor point), the arrows indicate that the system is attracted to this point, but if a perturbation occurs near the white circle (unstable or repelled point) the arrows indicate that the system goes some where else different from this point.

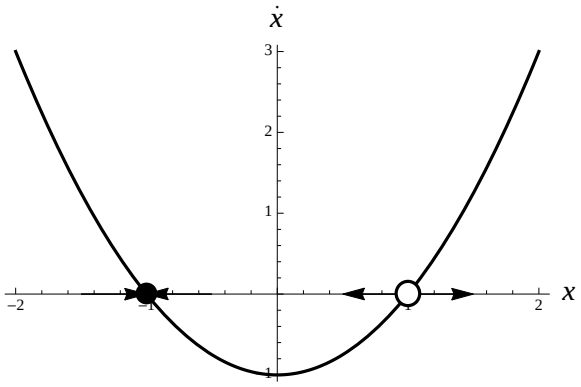


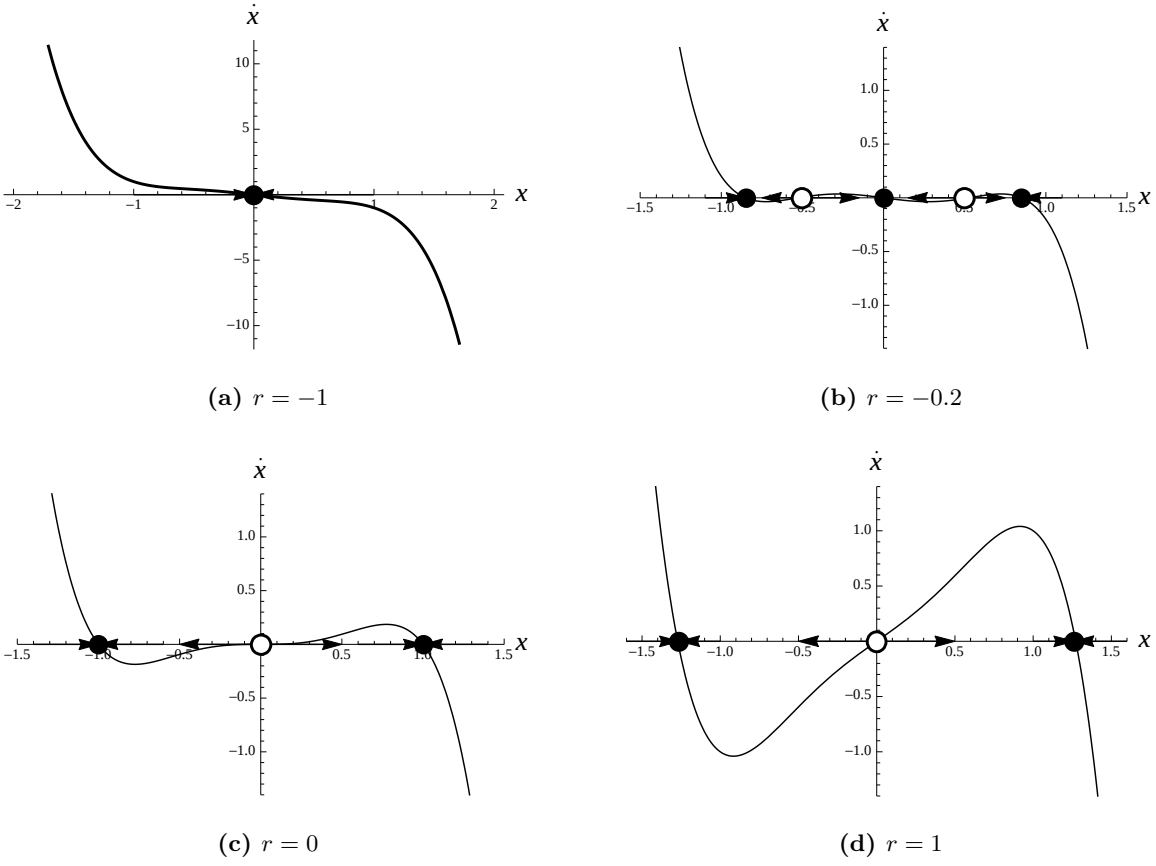
Figure 1.1: Unidimensional flow for a first-order system

In all fixed points the derivative is zero, but there is a difference between stable and

unstable equilibrium. We say that x^* is a stable fixed point if all trajectories that start near x^* approach it as $t \to \infty$. On the other hand, a fixed point x^* is unstable, if all trajectories that start near it are driven away from it.

First-order systems are very simple systems, but they exhibit interesting features when their parameters change with time. The qualitative structure of the phase portraits can change when we allow parameters to vary. Fixed points can be created or destroyed, or their stability can change. The qualitative changes in the topology of a phase portrait, due to the change in parameters are called *bifurcations*, and the value of the parameters where a bifurcation occurs are called *bifurcation points*.

Figure 1.2 shows the flow plots for equation $\dot{x} = rx + x^3 - x^5$, for different values of parameter r; Figure 1.3 shows the respective Bifurcation Diagrams (BD).



(a) $r = -1$ **(b)** $r = -0.2$

(c) $r = 0$ **(d)** $r = 1$

Figure 1.2: Flow plots of $\dot{x} = rx + x^3 - x^5$ for different values of r

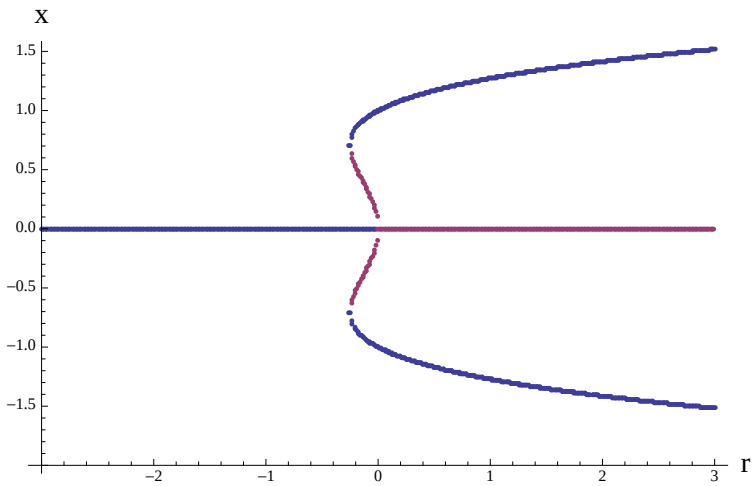


Figure 1.3: Bifurcation diagram for $\dot{x} = rx + x^3 - x^5$. The black points corresponds to stable fixed points, and the white points to unstable fixed points.

Bifurcation analysis has a large number of applications in science and engineering. Those applications range from radiation in lasers, outbreaks in insect populations, electronics, electrical power systems, etc. [Strogatz94].

In order to derive the behavior of the system in the presence of bifurcations, it is necessary to create a qualitative representation of the bifurcation diagram. This qualitative representation must capture all the essential characteristics of the bifurcation diagram. We use the qualitative representation to determine the behavior of the system.

1.2. Main Objective

Describe the qualitative behavior of a dynamical system from its quantitative bifurcation diagram when one of its parameters is varied.

1.2.1. Particular Objectives

- Creates a qualitative representation of a bifurcation diagram that captures all the essential information of the bifurcation diagram.
- Creates an algorithm that generates the qualitative representation from its bifurcation points.

- Creates an algorithm that describes the behavior of the system when one of its parameters is varied, using the qualitative representation.
- Creates an algorithm that describes the changes needed to one parameter of the system in order to take the system from an initial state to a final state.
- Integrate the three algorithms described before to create a prototype application.

1.3. Justification

In this thesis I do not attempt to produce the bifurcation diagram from its differential equation. I use the bifurcation diagram obtained from analytic, numerical or by other alternatives like the ones developed by Flores and Lopez [Flores10] or by Barrera and Flores [Barrera08]. Also in some of our experiments I used the information obtained from XPPAUT [Ermentrout03] .

Textbooks like Nonlinear Dynamics and Chaos by Strogatz [Strogatz94] describe the dynamic of a system using only the qualitative information of its derivatives. Using these principles, I apply qualitative reasoning to describe the qualitative behavior of a dynamical system. Using qualitative reasoning to describe the behavior of a physical system it is necessary to capture the essential features of the physical system and translate them into a qualitative representation. And create new methods that describe the behavior of the physical system. In the thesis I produce the qualitative representation of the BD and implement some algorithms that derive the possible behavior of the system.

With the implemented algorithms placed all in one single application called QBD it is possible for researchers to predict the behavior of a dynamical system in an easy way than the traditional one.

1.4. Previous Work

Earlier research in qualitative reasoning developed qualitative representations and reasoning algorithms that provide solutions in qualitative terms. In 1984 de Kleer and Brown introduced the concept of confluences in the article Qualitative Physics Based on

Confluences [deKleer84]. Forbus in 1984 worked on what he called Qualitative Process Theory[Forbus84], in 1986 Kuipers worked on Qualitative Differential Equations in the book Qualitative Reasoning [Kuipers94]. In 2000 he also presented the work Semi Qualitative System Identification [Kay00]. All these works use qualitative reasoning to solve a particular problem; we took some ideas from them to propose our qualitative representation of a bifurcation diagram and developed our qualitative simulation and planning algorithms. Also works like the ones presented by Lee[Lee93] and Sacks [Sacks90] dealt with dynamical systems and performed some kind of qualitative reasoning with phase portraits. Nevertheless, none of them dealt specifically with bifurcations or performed any qualitative reasoning based on bifurcation diagrams. Currently there is no research work in this direction, or a close approach.

In one related article, Yip's approach deals with discrete dynamical systems and generates phase space portraits by intelligently guiding the numerical simulations performed to complete the phase portrait with as many features as possible [Yip87]. Yip mentions that bifurcations are detected when they notice a change in the phase portrait; however, neither the bifurcations, bifurcation points, nor bifurcation diagrams are modeled or used in reasoning.

The qualitative reasoning system developed here, called QBD (for Qualitative Bifurcation Diagrams), is based on the earlier work of Flores et al. [Flores06]. Their reasoning system started from a given, qualitative representation of a dynamical system and performed a qualitative simulation in order to obtain a prediction of the system's behavior. QBD extends Flores et al.'s work in several ways. First, the qualitative representation of bifurcation diagrams has been improved; the new representation helps to solve the planning problem, where in its previous version [Flores06] could not solve. In particular, the inclusion of the transition table, which indicates to what state the system will go in response to different external events (i.e., perturbation or changes of the parameter), facilitating the simulation and planning algorithms. The resulting transition graph is a kind of envisionment [Johan De Kleer84] of possible trajectories the system can traverse in the bifurcation diagram. Second, QBD extends the earlier work by automating the translation of a quantitative bifurcation diagram into a qualitative form. Third, the simulation of the impacts of

system events and control actions on qualitative system behavior is performed by a more robust and improved algorithm. Finally, the development of control plans that can create desired changes in a qualitative system's state is a completely new feature. At present, the qualitative representation and the three algorithms presented in this thesis work with dynamical systems of one parameter.

1.5. About this Document

The present document is organized as follow: Chapter 2 describes the basic concepts about dynamical systems, bifurcation diagrams, and stability in bifurcation diagrams. Chapter 3 proposes a qualitative representation that takes the essential features of a bifurcation diagram. This representation will be used for the simulation and planning algorithm. Also, Chapter 3 presents an algorithm that generates the qualitative representation of the bifurcation diagram proposed in the previous chapter. Chapter 4 presents the simulation and planning algorithms. Both algorithms describe the behavior of the system in a qualitative way. Some examples are described in Chapter 5. Finally the conclusions are presented in Chapter 6.

Chapter 2

Dynamical Systems

2.1. Introduction

Dynamics is the study of change. A dynamical system describes how the variables of a system interact with each other and change with time. From a different perspective, a dynamical system is a concept in mathematics where a fixed rule describes the time dependence of a point in a geometrical space. Dynamical systems are mathematical objects used to model physical phenomena whose state changes over time. These models are used in power systems, financial and economic forecasting, environmental modeling, medical diagnosis, industrial equipment diagnosis, and a host of other applications.

Dynamical systems are divided in deterministic and stochastic. A deterministic system is a system in which no randomness is involved in the development of future states of the system. If the initial state of a dynamical system is known exactly, then the future state of the system could theoretically be predicted. Contrary, stochastic means *pertaining to chance*, therefore for a stochastic system one or more parts of the system has randomness associated with it. In other words, a stochastic system does not produce the same output given a particular input [Boukas02].

To illustrate a dynamical system, we could use a fish population example. Consider x_k is the current population of fish, and x_{k+1} is the next year's population. Given this nomenclature we could say that $x_{k+1} = F(x_k, x_{k-1}, ..., x_0)$ for any year k.

In this Chapter we present a brief history of dynamical systems, classification, and basic concepts of dynamical systems. This chapter also provides an introduction of concepts like fixed point, bifurcation diagram, and the different types of bifurcation that occur in a dynamical system.

2.2. Brief History of Dynamical Systems

Although at first dynamics was a branch of physics, with time it became an interdisciplinary subject. The study of dynamics starts in the middle 1600s, when Newton invented differential equations, while discovering the motion and gravitational laws. Both laws (motion and gravitational) combined, explained Keplers' laws of planetary motion [Russell64]. In particular, he solved the two body problem (i.e., calculate the motion of earth around the sun) [Strogatz94].

At the end of 1800s Poincaré [Poincaré85] initiated the qualitative theory of differential equations, by introducing a new point of view that emphasize qualitative rather than quantitative questions. One of the big questions, unresolved in general to this day, is the long-term stability of something like our solar system. Poincaré used his qualitative theory of differential equations to study this problem [Farkas81]. He was the first person to introduce the possibility of chaos, where a deterministic system exhibits aperiodic behaviors that depend particularly on its initial conditions [Wiggins90]. But chaos was not taken as something relevant in the first half of 1900s, instead researches focused on nonlinear oscillators and their applications to physics and engineering [Lakshmanan96]. Nonlinear oscillators played and important role at the time of development of new technologies like radio, radar, lasers, etc.

The invention of a high-speed computer in the 1950's allowed scientists to experiment with differential equations in a way that was impossible before. Such experiments led Lorenz in 1963 [Lorenz63], to discover a chaotic motion on a strange attractor. He found that the solution of these differential equations never settled to an equilibrium state, the system kept oscillating in an irregular way [Yang02]. Under those conditions Lorenz showed the existence of structure in chaos. When he plotted the solution in three dimensions, the

set of points took the form of a butterfly; we recognize a fractal structure in that attractor [Peitgen04].

The boom of chaos took place until the 1970's. In 1971 Ruelle and Takens proposed a new theory for the onset turbulence in fluids [Ruelle70], based on abstract considerations about strange attractors. Years later, May found examples of chaos at the time of interaction of arising population maps. Next Feigenbaum [Ivancevic08] discovered that there are certain universal rules governing the transition from regular to chaotic behavior. His work established the link between chaos and phase transitions [Strogatz94].

In 1970s Mandelbrot codified and popularized fractals, and producing magnificent computer graphics of them [Mandelbrot04], Winfree applied geometric methods of dynamics to biological oscillations especially circadian and heart rhythms [Winfree80].

2.3. Discrete and Continuous Systems

There are two main types of dynamical systems: differential equations and iterated maps. Differential equations describe the evolution of systems that are continuous in time, whereas iterated maps arise in problems where time is discrete. Discrete time systems view values of variables as occurring at distinct, separate points in time. Thus a variable jumps from one value to another as time moves from one time period to the next. In these systems, each variable of interest is measured once at each time period. The number of measurements between any two time periods is finite. Measurements are typically made at sequential integer values of the variable time.

In contrast, continuous time systems view variables as having a particular value for potentially only an infinitesimally short amount of time. Between any two points in time there are an infinite number of other points in time. The variable time ranges over the entire real number line, or depending on the context, over some subset of it (e.g the non-negative reals).

2.3.1. Discrete Systems

Discrete dynamical systems can be used to model and analyze many real-world problems including population growth, compound interest and annuities, radioactive decay, pollution control, and medication dosages.

When we model a system as a discrete dynamical system, we imagine that we take a snapshot of the system at a sequence of times. The snapshots could occur once a year, once every millisecond, or even irregularly, such as once every time a new government is elected. To complete the description of a dynamical system, we need to specify a rule that determines, given an initial snapshot, what the resulting sequence of future snapshots must be.

Informally, a discrete dynamical system is a numerical succession, defined as a relation of recurrence. We mean that there is a rule that defines every number of the succession based on the previous ones. For example if we open a bank account with an initial amount of 100 and with an interest rate of 6 %, the next state will be the next year because the interest rate is annual. With this information we can describe the rule of change as Equation (2.1).

$$X_{k+1} = 1.06X_k \tag{2.1}$$

A discrete system can also be considered as a discrete representation of a continous system.

2.3.2. Continuous Systems

In the last example the change occurs annually, nevertheless, there are systems that are constantly changing, therefore the time can not be expressed as intervals. Generally in discrete systems k denotes discrete time points, now for continuously changing problems we use t instead of k to denote time, where t is a non negative real number instead of an integer.

For example, if we throw a ball vertically, the system can be described at any instant of time by two real numbers, $h(t)$ to define the height of the ball and $v(t)$ to define

its speed. If we know that the speed is defined by $dh(t)/dt = v$ and gravity is a factor that affects the ball, defined by $d^2h(t)/dt^2 = -g$. Given this, the change in the system can be defined in the Differential Equation (2.2).

$$\begin{aligned} \dot{h}(t) &= v(t) \\ \dot{v}(t) &= -g \end{aligned} \tag{2.2}$$

2.4. Linear and Nonlinear Systems

The term linear is used in geometry to specify objects like lines or planes. This kind of objects keeps the same shape at the time of changing the scale that is been seen. An example of nonlinear object is a sphere. This kind of objects may have a different appearance, depending on the scale used to observe it. For example, if you look at the sphere from too close, in the limit, looks like a plane. Instead, if you look at it from too far, it could look like a point.

2.4.1. Linear Systems

Linear dynamical systems are dynamical systems where all functions that compose them are linear. While dynamical systems in general do not have closed-form solutions, linear dynamical systems can be solved exactly. In mathematics, a linear function f(x) is a function which satisfies the following three properties [Farina11]:

- **Additivity**: $f(x + y) = f(x) + f(y)$.
- **Homogeneity** of degree 1: $f(\alpha x) = \alpha f(x)$ for all α.
- **Superposition**: $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$.

Linear systems can also be used to understand the qualitative behavior of general dynamical systems, by calculating the equilibrium points of the system and approximating it as a linear system around each such point.

2.4.2. Nonlinear Systems

A system is non linear, if it does not fulfill all the proprieties of linear systems. Usually nonlinear functions represent a big issue to solve a system. Some times takes to much time to find the solution or it cannot be solved. The range of behaviors obtained in the solutions of nonlinear systems is much wider than those found in linear ones. That is the reason because there has been some research to solve them [Strogatz94].

2.5. Stability

Stability theory addresses the stability of solutions of differential equations and the trajectories of dynamical systems under small perturbations. A system is stable if at the time of small perturbations to the system, the system does not diverge. Contrary, the system is unstable if, the system diverges when there is a small perturbation. For example, Figure 2.1 shows two kinds of surfaces; if we apply a small perturbation to the mass in Figure 2.1 (a) it will return to its original position (i.e., it represents a stable point). On the other hand, if we push slightly the mass of Figure 2.1 (b), it will go away from its original position (i.e., it represents an unstable point).

Figure 2.1: Stability

2.6. Geometry

We consider the system defined by the differential equation (2.3)

$$\frac{dx}{dt} = sin(x) \tag{2.3}$$

To obtain the analytic solution it is necessary to separate the variables and integrate, as shown in Equation (2.4)

$$\begin{aligned} dt &= \frac{dx}{sin(x)} \\ t &= \int csc(x)dx \\ t &= -ln|csc(x) + cot(x)| + C \end{aligned} \tag{2.4}$$

To evaluate constant C, lets us assume that $x = x_0$ at $t = 0$. Then

$$C = ln|csc(x_0) + cot(x_0)| \tag{2.5}$$

Substituting Equations (2.4) and (2.5) we obtain Equation (2.6)

$$t = ln|\frac{csc(x_0) + cot(x_0)}{csc(x) + cot(x)}| \tag{2.6}$$

The result (Equation 2.6) is exact but not easy to interpret. On the contrary, a graphical analysis of Equation 2.3 is clear and simple as shown in Figure 2.2. Think t as time, x as the position of a particle moving along the real line, and $\dot{x}$ as the velocity of that particle. The differential equation $\dot{x} = sin(x)$ represents a vector field on the line. Figure 2.2 shows graphically the fixed points and their stability of Equation (2.3). The arrows indicate the flow of the system. The arrows point right if $\dot{x} > 0$, $\dot{x} < 0$ to the left, and when $\dot{x} = 0$ there is no flow. This points are call fixed points.

Figure 2.2 shows two kinds of fixed points. Black points represent stable fixed points, also call attractors because the flow is oriented to them. Circles represent unstable ones, also call repellers.

For example, to define the fixed points and stability of a system defined by the differential equation $\dot{x} = x^2 - 1$, it is necessary to find when the slope is zero ($\dot{x} = 0$). After solving we found the solution at $x = \pm 1$. We plot the the graph of $x^2 - 1$ (Figure 2.3) and mark the fixed points. According with the flow of the graph, we found that the fixed point at $x = -1$ is stable and the one at $x = 1$ is unstable. Figure 2.3 shows the fixed points of the system and their stability.

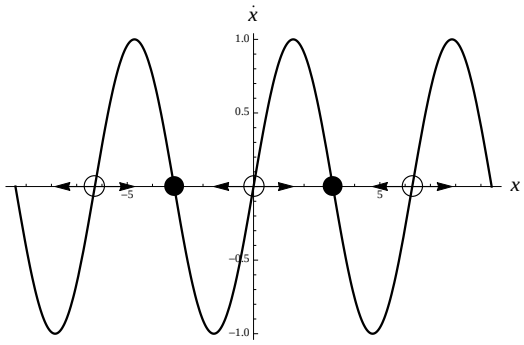


Figure 2.2: Fixed points of the system described by the Differential Equation (2.3).

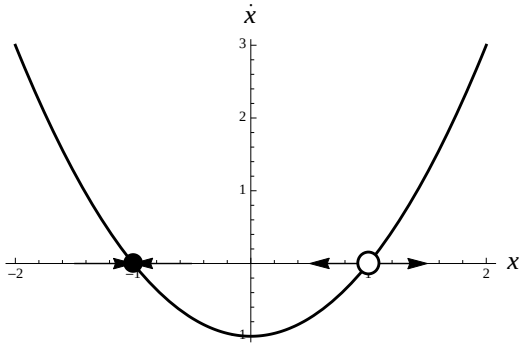


Figure 2.3: Fixed points and stability of the system described by the Differential Equation $\dot{x} = x^2 - 1$.

2.7. Bifurcations

Bifurcation theory is a subject with a classical mathematical origin; however, the modern development of the subject starts with Poincaré and the qualitative theory of differential equations [Farkas81]. In recent years, this theory has undergone a tremendous development with an infusion of new ideas and methods from dynamical systems theory [Crawford91].

A bifurcation occurs in a dynamical system when at the time of varying one of its parameters there is a qualitative change in the behavior of the system. A bifurcation point is defined by the values of the parameters where a bifurcation occurs. For example, let us consider the buckling of a beam. If a small weight is placed on top of a beam, the beam can support the load, but if the load starts increasing constantly, there will be a time that the

beam may not support the load. At the time that the vertical position becomes unstable, the load may buckle and a bifurcation in the system occurs. Figure 2.4 shows the buckling of a beam.

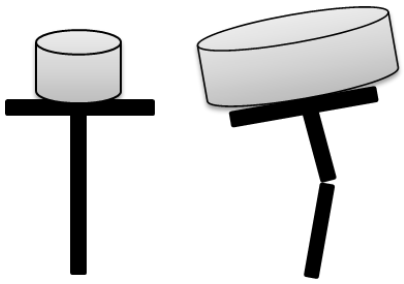


Figure 2.4: Buckling beam.

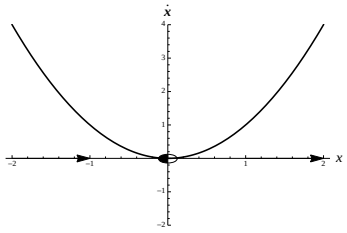
2.7.1. Saddle-node Bifurcation

The saddle-node bifurcation is a phenomenon present in dynamic systems where fixed points are created and destroyed. A saddle-node bifurcation occurs when two fixed points, one stable and the other one unstable, approximate until they meet. The saddle-node point is the limit point between the stable zone and the unstable one. For example, consider the system defined by the Differential Equation (2.7).

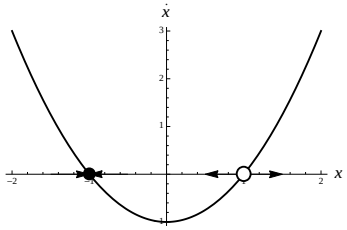
$$\dot{x} = r + x^2 \tag{2.7}$$

where, r is the system's parameter. When r is negative, there are two fixed points, one stable and the other one unstable. When r approaches zero, both points approximate each other; so at the time that r is zero, both points are at the same point giving place to the saddle node bifurcation. At this point, form one side the system is unstable, and from the other one it is stable. If $r > 0$ there are no fixed points. This example can be observed graphically in Figure 2.5.

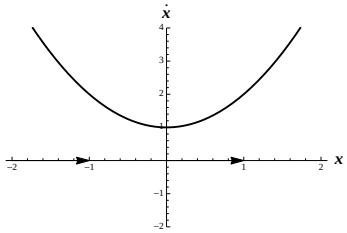
A bifurcation diagram is a graphical expression that includes the fixed points of the system for all possible values of the system parameter r. For this example, the bifurcation diagram can be observed in Figure 2.6. Figure 2.6 the dashed section of Figure 2.6 represents unstable fixed points, and the continuous line represents the stable fixed points.



(a) $r = 0$



(b) $r < 0$



(c) $r > 0$

Figure 2.5: Behavior of $\dot{x} = r + x^2$ for different values of r.

2.7.2. Transcritical Bifurcation

There are certain situations where a fixed point must exist for all values of a parameter and can never be destroyed. For example, in simple growth models of a single species, there is a fixed point at zero population, regardless of the value of the growth rate. However, such fixed points may change their stability as the parameter is varied. The transcritical bifurcation is the standard mechanism for such changes in stability [Strogatz94].

The normal form[1] for a transcritical bifurcation is defined in Equation (2.8).

[1] A normal form is the simplest differential equation that captures the essential features of a system what exhibitsa certain type of bifurcation.

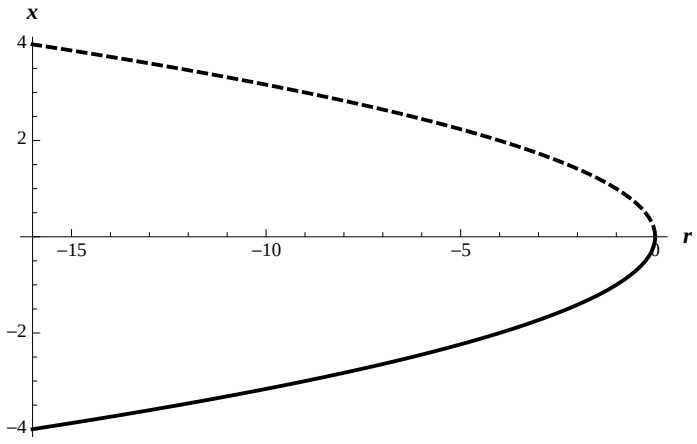


Figure 2.6: Bifurcation Diagram that illustrates a Saddle-node Bifurcation. The system is defined by the Differential Equation (2.7).

$$\dot{x} = rx - x^2 \tag{2.8}$$

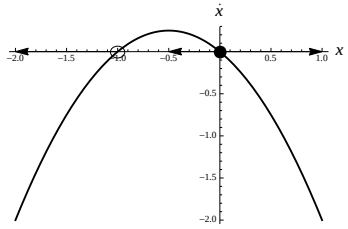
Figure 2.7 shows that regardless of the value of parameter r, there always exists at least one fixed point. For example, when $r < 0$ there is an unstable fixed point at $x = r$ and a stable fixed point at $x = 0$; when $r > 0$ exists an unstable fixed point at $x = 0$ and a stable one at $x = r$. And when $r = 0$ there is a semi-stable fixed point at $x = 0$. Note that an exchange of stability has occurred to the fixed point at the origin.

The difference between the saddle-node and the transcritical bifurcation is that in the last case, the fixed point never disappears after a bifurcation, it just changes stability. Figure 2.8 shows the bifurcation diagram of the system.

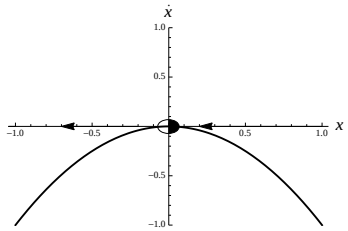
2.7.3. Pitchfork Bifurcation

Pitchfork bifurcations are common in physical problems that have symmetry, where the fixed points appear and disappear in pairs. There are two kinds of bifurcations for this type, the supercritical and the subcritical [Strogatz94].

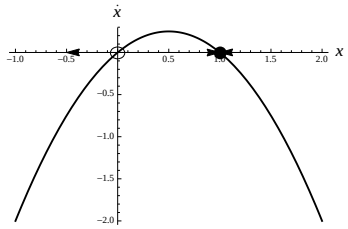
The normal form of supercritical pitchfork bifurcation is defined by Equation (2.9). Note that this equation is invariant under the change of variables $x \to -x$. If we replace x by $-x$ and then cancel the resulting minus signs on both sides of the equation, we get the



(a) $r < 0$



(b) $r = 0$



(c) $r > 0$

Figure 2.7: Behavior of $\dot{x} = rx - x^2$ for different values of r.

initial equation. Figure 2.9 shows the behavior of the system when the parameter r changes.

$$\dot{x} = rx - x^3 \tag{2.9}$$

When $r \leq 0$, the origin is the only stable fixed point which is stable. If $r > 0$ the origin is unstable, and two new stable fixed points appear symmetrically located at $x = \pm\sqrt{r}$. Figure 2.10 shows graphically the bifurcation diagram of the system defined by Equation (2.9).

The normal form for the subcritical pitchfork bifurcation is given by Equation (2.10). Note that the difference between the supercritical is the cubic term. In the supercri-

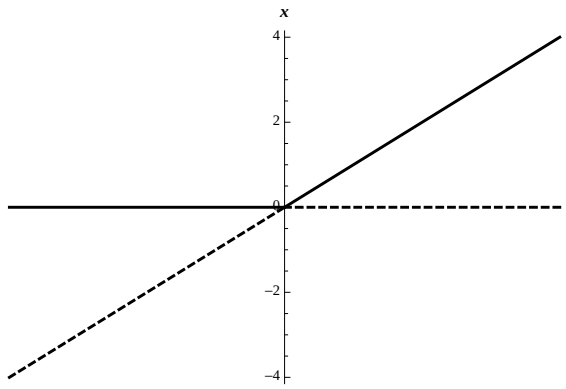


Figure 2.8: Bifurcation Diagram of the system defined by the Differential Equation (2.8)

tical case, the negative sign of the cubic term acts as a stabilizer, now being positive, it is destabilizing.

$$\dot{x} = rx + x^3 \tag{2.10}$$

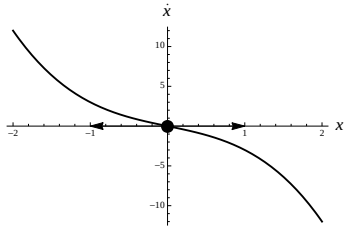
Figure 2.11 shows the bifurcation diagram for the system defined by Equation (2.10). Compared with the supercritical bifurcation, now the pitchfork is inverted. The fixed points that are located over $x = \pm\sqrt{r}$ are unstable, and the origin is stable when $r < 0$, and unstable when $r > 0$.

2.7.4. Jumps and Hysteresis

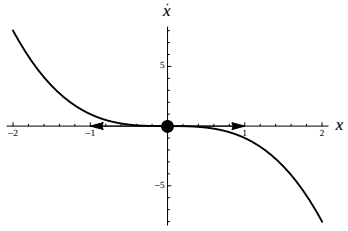
To explain jumps and hysteresis let us consider the system defined by the Differential Equation (2.11).

$$\dot{x} = rx + x^3 - x^5 \tag{2.11}$$

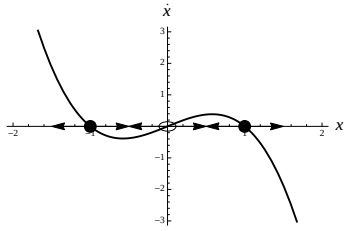
The bifurcation diagram of the system is shown in Figure 2.12. From the bifurcation diagram we can observe that the origin is locally stable when $r < 0$. Two symmetric branches emerge from origin until $r = r_s$. At $r = r_s$, two new branches emerge, switching stability from unstable to stable for all $r > r_s$.



(a) $r < 0$



(b) $r = 0$



(c) $r > 0$

Figure 2.9: Behavior of $\dot{x} = rx - x^3$ for different values of r.

Let us analyze the bifurcation diagram of the Figure 2.12, and consider the following:

1. The range of $r_s < r < 0$, two qualitative different stables states coexist.

2. The existence of different states allows the possibility of jumps and hysteresis when the parameter r is being varied. For example, lets consider the initial state of the system at $r < r_s$. At the time of increment for the parameter r, the system remains at the origin until $r = 0$. When $r \geq 0$, the origin became unstable; now a small perturbation will cause that the system jumps to one of the large-amplitude branches. If we continue incrementing the value of the parameter r, the system will remains in the same branch.

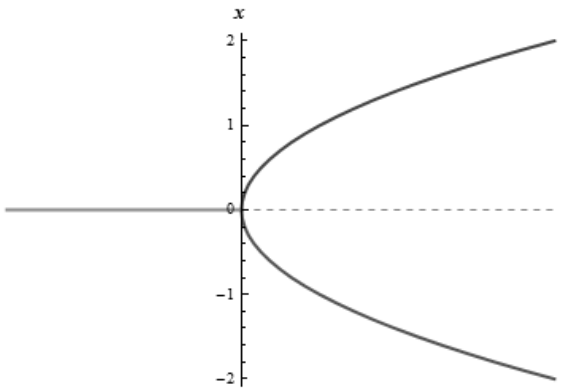


Figure 2.10: Supercritical Pitchfork Bifurcation Diagram of the system defined by the Differential Equation (2.9).

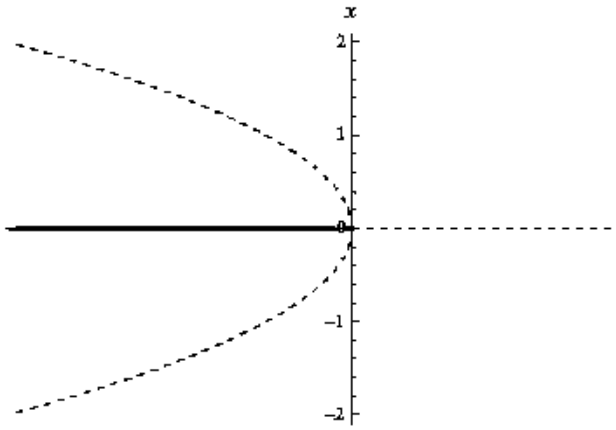


Figure 2.11: Supercritical Pitchfork Bifurcation Diagram of the system defined in the Differential Equation 2.10

Furthermore, if we decrement the value of the parameter r lower than r_s, the system jumps one more time to the origin. This lack of reversibility as a parameter changes is called hysteresis. Figure 2.13 shows this example of jumps and hysteresis.

3. The bifurcation at r_s is a saddle-node bifurcation, where two fixed points (one stable and the other unstable) appear as r increases.

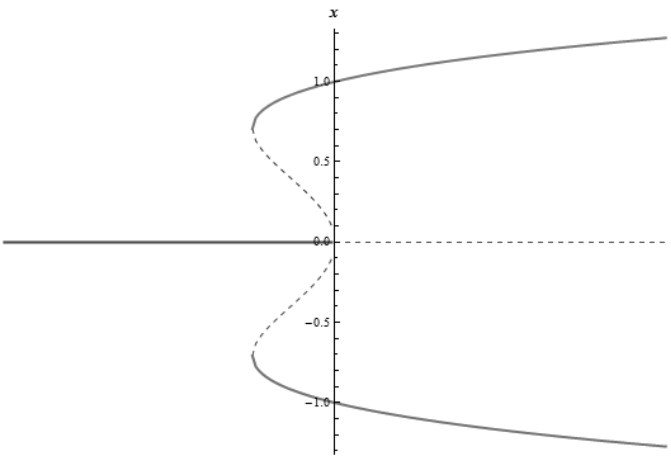


Figure 2.12: Bifurcation Diagram of the system defined in the Differential Equation (2.11).

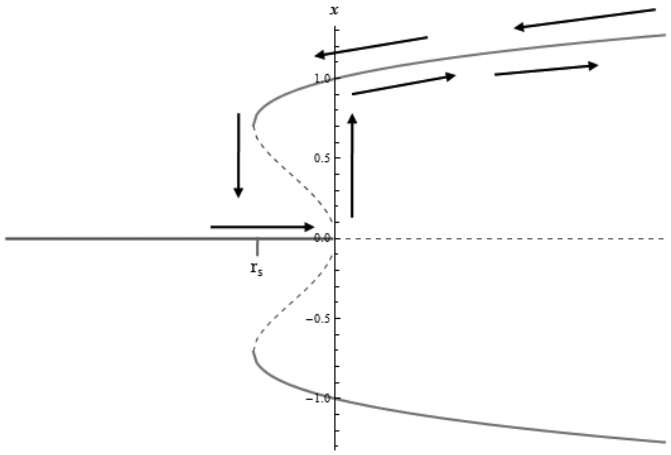


Figure 2.13: Example of jumps and hysteresis

2.8. Chapter Conclusions

This chapter introduced some basic concepts about dynamical systems. These concepts include fixed point, stability, and bifurcation diagram, which will be used in the following chapters. In the following chapters we will propose a qualitative representation of a bifurcation diagram. With the qualitative representation, we present two algorithms that describe the behavior of a dynamical system when one of its parameters changes.

Chapter 3

Qualitative Representation of a Bifurcation Diagram

Qualitative reasoning is the area of artificial intelligence that creates approximate representations of continuous aspects of the world, such as space, time, and magnitude, and supports reasoning with as little exact information as possible. The goal of qualitative reasoning is to arrive at useful conclusions about a system when precise information about the system is not available or is not necessary. This information about the system corresponds to a model of the system (the numerical information of the system may or may not be known). The problem is to select the essential characteristics of the system and put them in a useful representation.

To reason about a situation or a system, it is usually necessary to model the system. To create a model, it is essential to identify the relevant objects of the system and their relationships within the system. Besides identifying the relevant objects, it is mandatory to determine their essential characteristics. Once the identification of the essential objects and their characteristics is performed, it is necessary to put this information into a given representation.

A qualitative representation of a system is a description of the system' essential characteristics. It is obvious that there is no single, right, or best qualitative representation. Instead, there exists a spectrum of choices, each with their own advantages and disadvanta-

ges for a particular task. The characteristics that all of them have in common is that they provide a notation for describing and reasoning about continuous properties of the physical world in a discrete fashion.

Forbus in [Forbus97] proposes that two key aspects of a qualitative representation are its resolution and its compositionality. Resolution concerns the level of information detail in a representation, which is an issue because one goal of qualitative reasoning is to understand how little information is sufficient to draw useful conclusions.

Compositionality concerns the ability to combine representations for different aspects of a phenomenon or system to create a representation of the phenomenon or system as a whole. Compositionality is an issue because one goal of qualitative physics is to formalize the modeling process itself.

For our problem of capturing the essential characteristics of the bifurcation diagram for the qualitative representation, we decompose the diagram in sections with common qualitative characteristics. The common qualitative characteristics in this work are a set of bifurcation points with the same derivative and same behavior (stability). In the following sections we explain the structure of the proposed qualitative representation and we present an illustrative example of it.

3.1. Qualitative Representation

The main components of a dynamical system are variables, parameters, and constraints[1]. Since we are starting from a bifurcation diagram we do not need to represent the structure of the system (the constraints). A bifurcation can thus be described in terms of the values of its state variables and parameters at different points in time. At any point in time, the value of each variable or parameter is specified in terms of its relationship with a totally ordered set of landmark values and its direction of change. The set of landmark values is called a quantity space.

The qualitative value of a variable can be a landmark or an open interval between landmarks. The qualitative direction of a variable is the sign of its derivative, in this case,

[1]The constraints of a dynamical system are generally represented as a set of Ordiary Differential Equations.

related to change with respect to another variable. A landmark is a symbolic name for a particular qualitative value (whose numerical value may or may not be known) that breaks a continuous set of values into regions with different qualitative characteristics.

In a Bifurcation Diagram (BD), we plot the system's state variable x, versus a parameter r. Generally, x is not a function of r (in the strict mathematical sense), so we need to represent the bifurcation diagram as a relation.

The qualitative representation of a bifurcation diagram, a QBD, is a tuple of four elements $\langle V, QS, BDS, T\rangle$. Where V is defined the state variable and the parameter variable. QS defines the sets of state variable and parameter landmaks in the system. BDS defines the Monotonic Segments (MS), each MS is described by its initial and final points, its nature and its direction. T defines the transitions or next state for every possible action (i.e., perturbation or increase/decrease in the parameter) for every MS defined in BDS. The qualitative representation syntax is defined using EBNF (Extended BackusNaur Form) [Scowen93].

- V is a set of the variables of the dynamical system, where x correspond to the state variable and r to the parameter variable. Note that this representation only works with bifurcation diagrams of one state variable and one parameter.

  $V = (x, r)$

- QS is the set of the qualitative quantity spaces, one for each variable in V. Each quantitative space has a set of landmarks; consider there are n_x for the variable x and n_r landmarks for r. Notice that all numerical values are discarded, landmarks are only ordered by magnitude. This mean that $x_{k-1} < x_k < x_{k+1}$.

  $QS = (QS_x, QS_r)$

  $QS_x = (x_1, x_2, x_3, .., x_{n_x})$

  $QS_r = (r_0, r_1, r_2, ..., r_{n_r})$

- BDS is a set of n monotonic segments that defines the bifurcation diagram. Every MS is described by its initial and final points, its nature and its direction. The initial and final points are defined by two landmarks, one contained in QS_x and other in QS_r.

Nature is the stability of the segment (s for stable and u for unstable), and direction is the slope (i.e., -, 0, +).

$BDS = (S_1, S_2, .., S_{n_s})$

$S_i = ((x_a, r_b), (x_c, r_d), (nature, direction))$

$nature = s \mid u$

$direction = - \mid 0 \mid +$

- T is a set of transitions; each segment's transitions has four possible directions (Up, Down, Left and Right). Two types of transitions are defined, the first case is when a perturbation occurs. If a perturbation occurs, the Up or Down transition defines the possible segments (if they exist) that the system will arrive to in case of a positive (Up) or Negative ($Down$) perturbation. There are n_{pert} perturbations transitions defined for every segment for positive and negative perturbations. The segment that the system will arrive to depends on where in the BDS the perturbation took place. That is the reason why a segment may have more than one transition; in such cases the boundary ($Limits$) of $Pert$ must be defined (i.e., (r_{init}, r_{end}).

  The second case occurs when the parameter r changes. If r increases and the value of r goes over the limits of the segment, the system transits to another segment. Left and Right Transitions define the the next segment (if it exist) that the system will arrive to when the state reaches a point beyond the segment boundary. Left transition for the case of going beyond the left end and Right Transition when going beyond the right end. Notice that for this case there is only one possible segment that the system may arrive to (See Flores work in [Flores06]).

  $T = (Up\ Transitions, Down\ Transitions, Left\ Transitions, Right\ Transitions)$

  $Up/Down\ Transitions = \{Pert_1, Pert_2, ..., Pert_l\}$

  $Pert = (S_{next}, Limits)$

  $Limits = (r_{init}, r_{end})$

  $Left/RightTransitions = S_i$

In summary a QBD is comprised of a set of qualitative segments (BDS) of a given bifurcation diagram. Each segment is considered to be a qualitative line; a line has beginning and end points, a qualitative slope or *direction* (i.e., +, -, or 0), and in this case, each line has its own *nature* (i.e., is stable or unstable). Each segment may have segments beyond both of its ends (Left or Right). The transitions of each segment (T) define the next segment that the system will follow in case of an event or action (i.e., perturbation, derailment, fall off, etc.). $Up/Down\ Transitions$ define the possible segments (if they exist) that the system will arrive to in case of a positive (Up) or Negative ($Down$) perturbation. *Pert* defines the next segment in case of a perturbation. For the case of a perturbation occurs in a stable segment ; Pert is the same segment where the perturbation occurs. Limits define the boundary of the perturbation. $Left/Right\ Transitions$ are used to define any possible segments that the system will transition to when the system has exceeded the $Left$ or $Right$ ends of the segment. With these directed connections between segments, a qualitative simulation can be produced by following paths through the model along segments and across transitions. These explicit representations of transitions extend the earlier representation [Flores06].

3.2. Example of the Qualitative Representation

To illustrate the proposed qualitative representation we use the system defined by $\dot{x} = 16 + rx - x^3$. For this example we use the application created by Flores and Lopez [Flores10] to create the bifurcation diagrams. Figure 3.1 shows the bifurcation diagram of the system; in the figure there are two kind of dots, blue dots correspond to stable fixed points ones and red dots correspond to unstable ones.

To perform qualitative simulation of the behavior of the system, it is necessary to capture only the essential characteristics of the diagram and translate them to an appropriate representation. In this case we need to group in sets of points with the same slope and stability. Figure 3.2 shows the set of points grouped by slope and stability. From Figure 3.2 we observe three sets of dots, the black dots correspond to stable fixed points with negative slope, the orange ones correspond to unstable fixed points with a positive slope, and the gray ones correspond to stable fixed points with a positive slope.

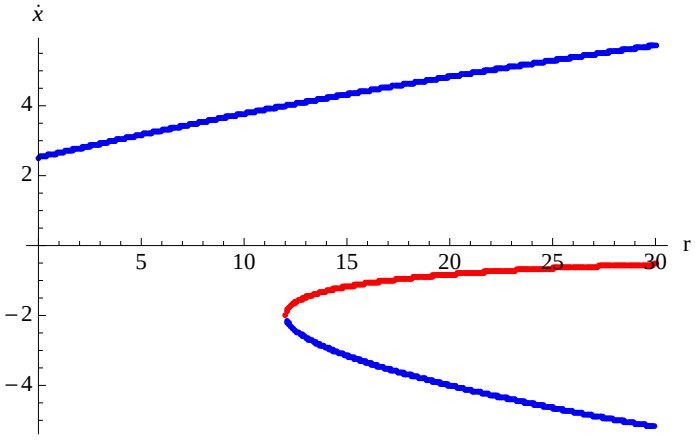


Figure 3.1: Bifurcation diagram defined by $\dot{x} = 16 + rx - x^3$. Blue dots correspond to stable fixed points and red dots to unstable ones.

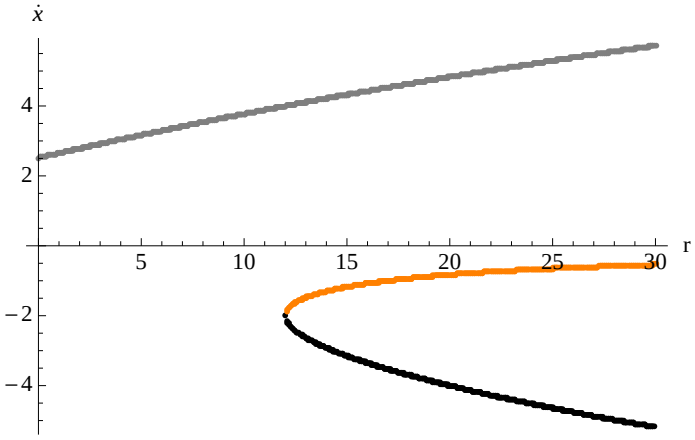


Figure 3.2: Breaking the bifurcation diagram of the system defined by $\dot{x} = 16 + rx - x^3$. The points are grouped by slope and stability.

If we take only the initial and final points of the previous sets and join them by a line we delete all the irrelevant information a take the essential characteristics. Figure 3.3 shows the monotonic segments of the bifurcation diagram; black dots represent stable segments and red ones represent unstable segments.

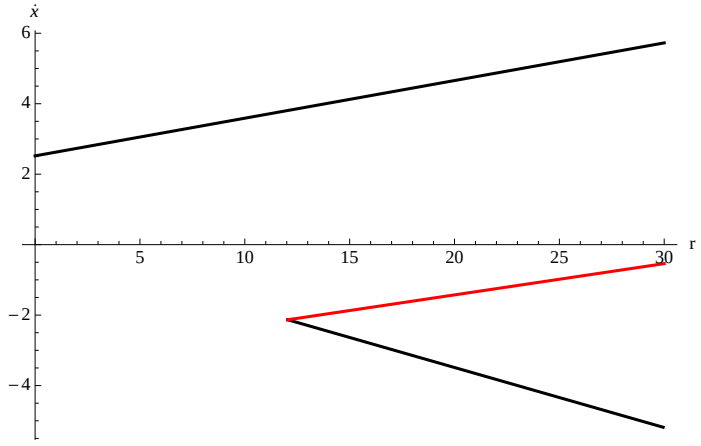


Figure 3.3: Bifurcation diagram of the system defined by $\dot{x} = 16 + rx - x^3$ divided in monotonic segments. The black lines correspond to a stable segment and the red line correspond to an unstable segment.

The qualitative representation of the BD of the Figure 3.2 is a tuple of four elements $\langle V, QS, BDS, T\rangle$. V is composed of x and r; x corresponds to state variable and r is the system parameter. The quantitative space for variable x is defined by four landmarks (x_0 to x_3) and for variable r by three landmarks (r_0, r_1,r_2). In the system there are only three qualitative segments each of them are defined by its initial and final point, its stability and its slope. Table 3.1 shows the qualitative representation of the system.

Transitions are defined in Table 3.2, where the directed connections between segments are defined. As we can observe in the Table 3.2 when a perturbation (positive or negative) occurs in a stable segment the system remains in the same segment (see the row of segment S_0 at column $Up/Down$ $=\{S_0, \{r_1, r_2\}\}$). But if the system is at S_0 and the system decrease longer than $r < r_1$, the system goes to segment S_2 (See row S_0 at column $Left = S_2$).

Finally if we plot the information contained in Table 3.1 we produce Figure 3.4. Figure 3.4 shows a qualitative representation of the bifucation diagram. The dotted lines

Table 3.1: Qualitative Representation of the Bifurcation Diagram of the system defined by $\dot{x} = 16 + rx - x^3$

Variables		$\{r, x\}$
Quantity Space	QS_r	$\{r_0, r_1, r_2\}$
	QS_x	$\{x_0, x_1, x_2, x_3, x_4\}$
BDS	S_0	$\{\{r_1, x_1\}, \{r_2, x_0\}, \{-, s\}\}$
	S_1	$\{\{r_1, x_1\}, \{r_2, x_2\}, \{+, u\}\}$
	S_2	$\{\{r_0, x_3\}, \{r_2, x_4\}, \{+, s\}\}$
Transitions		See Table 3.2

Table 3.2: Transitions of the Bifurcation Diagram of the system defined by $\dot{x} = 16 + rx - x^3$

Segment	Up	Down	Left	Right
S_0	$\{S_0, \{r_1, r_2\}\}$	$\{S_0, \{r_1, r_2\}\}$	S_2	$\{\}$
S_1	$\{S_2, \{r_1, r_2\}\}$	$\{S_0, \{r_1, r_1\}\}$	$\{\}$	$\{\}$
S_2	$\{S_2, \{r_0, r_2\}\}$	$\{S_2, \{r_0, r_2\}\}$	$\{\}$	$\{\}$

represent unstable segments and the continuous ones represent stable ones.

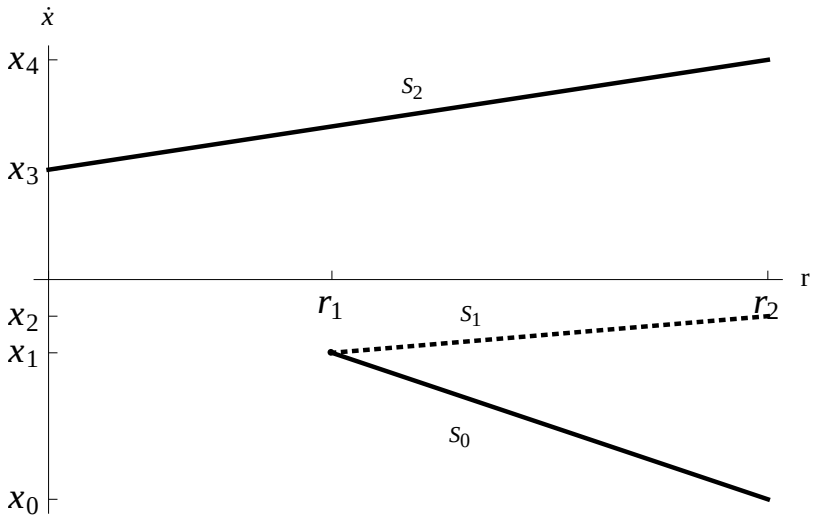


Figure 3.4: Bifurcation diagram of the system defined by $\dot{x} = 16 + rx - x^3$ divided in monotonic segments. The black lines correspond to a stable segment and the red line correspond to an unstable segment.

3.3. Semi-stable Fixed Points

A bifurcation diagram is a set of points that represent solutions of the differential equations under different conditions. There are stable, unstable, and semi-stable fixed points. (A semi-stable fixed point may be stable from above, or from below.) For instance, the points at the intersections of segments S_0 and S_1 of Figure 3.4 are fixed points where the derivative of the segment tends to infinity. Those fixed points are semi-stable; point p_1 is stable from above and unstable from below (see Figure 3.5).

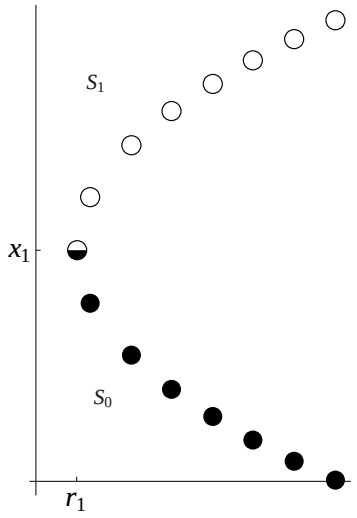


Figure 3.5: A Semi-stable Fixed Point

Since the qualitative nature of those points are different than those of their neighboring segments, unstable points would form segments by themselves. Given that the determination of a quantitative BD is done by a succession of discrete steps, the likelihood of capturing those points in the quantitative bifurcation diagram is very low. For the sake of completeness of the QBD, those semi-stable points could be included in the representation; nevertheless, their inclusion would not change the outcome of the qualitative simulation. The fact that the inclusion of semi-stable fixed points in the QBD is unnecessary will be illustrated in Section 4.2. By now, I want to make clear that those fixed points were considered and I decided not to include them in the qualitative representation.

As mentioned in this chapter there is no single, unique, or best qualitative repre-

sentation; there may be more than one of them . For our particular problem of predicting the behavior of a bifurcation diagram, we need to capture the essential characteristics of the system and translate them into an optimal qualitative representation. In this chapter we proposed a qualitative representation which includes as a tuple of four elements (variables, quantitative spaces, segments, and transitions). Chapter 3.4 describes an algorithm that generates the qualitative representation from its quantitative form (bifurcation points) and in Chapter 4 the simulation and planning algorithm are described. Both algorithms use the qualitative representation for qualitative simulations.

3.4. Qualitative Representation Generation

This section describes the algorithm that generates the qualitative representation of a bifurcation diagram. This algorithm, Qualitative Representation Generation (QRG) takes as input the quantitative representation of the BD (bifurcation points); i.e., the set of points from a quantitative bifurcation diagram along with the stability associated to each fixed point. Specifically, we accept files generated by the system of Flores and Lopez [Flores10] and XPPAUT [Ermentrout03], but our system could easily be adapted to use the output from other applications that generate dynamical system bifurcation diagrams.

It is possible to perform simulations to predict the behavior of the dynamical system based on the qualitative representation. Such a qualitative representation was proposed in previous section; the algorithm proposed in this section takes the essential characteristics of the system and translates them to the qualitative representation. *QRG* takes as input the bifurcation points, split them into subsets and then generates the monotonic segments (MSs). With these monotonic segments it takes the essential information to generate the qualitative representation.

3.4.1. Qualitative Generation Algorithm

The pseudo-code for the algorithm for Qualitative Representation Generation (QRG) is shown as Algorithm 1. Given the quantitative data of the bifurcation diagram, delta, DeltaGlue, and W; in Line 1 QRG proceeds to break the information into connec-

ted subsets, where the points appear to be part of a partial function (Section 3.4.2). A bifurcation diagram is usually not a function, but may have a number of operating regions for a given parameter value. In Line 3 each subset is then broken into smaller subsets, first regions and then *Monotonic Segments* (MS) (Section 3.4.3); all the points contained in an MS have the same qualitative behavior (i.e., sign of slope and nature). Since the input may have imprecision or errors, there can be small gaps in the information and, therefore, a lack of continuity between segments. Line 4 glues nearby segments together in order to reestablish the underlying continuity (Section 3.4.4). Finally, in Line 5 the algorithm takes the end points of the segments and store them with symbolic names (landmarks) in QS (Section 3.4.5). Once the qualitative segments of the bifurcation diagram have been defined, the transitions between segments (Section 3.4.6) are generated (in Line 6), determining behavioral links to other segments (Section 3.4.6). Last, the qualitative representation is generated (Line 7) and returned (Line 8).

Algorithm 1 `QRG`$(BDPoints, delta, DeltaGlue, W)$

1: $Subsets \leftarrow \{\}$
2: $Subsets \leftarrow$ `SegmentBifurcationDiagram`$(BDPoints, SubSets, delta)$
3: $MSs \leftarrow$ `BreakingInToMS`$(Subsets, MSs, W)$
4: $MSs \leftarrow$ `GluingSegments`$(MSs, DeltaGlue)$
5: $Landmarks \leftarrow$ `SettingLandmarks`(MSs)
6: $Transitions \leftarrow$ `GenerateTransitions`(MSs)
7: $QualitativeRepresentation \leftarrow$ `GenerateQR`$(MSs, Landmarks, Transitions)$
8: `Return` $QualitativeRepresentation$

3.4.2. Segmenting the Bifurcation Diagram

The segmentation algorithm takes as input the points in the quantitative bifurcation diagram ($BDPoints$). These points are generally not a function[2], and therefore we cannot treat them as such. It is useful to break the set of points into subsets, where all the points in a given subset are a function. To solve this problem, subsets are generated by taking all the points that can be observed from below (that is, taking the minimum value of x for every value of r). We start from the lowest point of the BD (Line 4). From that

[2] $f(x)$ is a function if for every value of x, the value of f is uniquely defined. In BDs, there are values of the parameter r for which x can be multiply defined (i.e., several fixed points may exist).

point we take all the points that can be observed from below first to the left (Line 5) then to the right (Line 6). During this process we discard the points that do not belong to the segment. To discard a point we check the distance of the point to its neighbors; *delta* is the parameter used to discriminate a point or not. For example, if we start from the lowest point of the diagram of Figure 3.6, and take all the points that can be observed from below, there are certain points that belong to another segment (when $r > 12$ the observed points are not being consider for the moment because the distance is greater than *delta*). If we do not make this distinction we could take points that do not belong to the segment. After selecting the points, they are removed from the original data (Line 8) and continue with the same process until no more points remain. Algorithm 2 shows the complete algorithm of this process.

Algorithm 2 `SegmentBifurcationDiagram`$(BDPoints, SubSets, delta)$

```
 1: if BDPoints is Empty then
 2:   Return SubSets
 3: else
 4:   Pos ← PositionLowestPoint(BDPoints)
 5:   Set1 ← SelectAllLeftPointsFromPos(BDPoints, delta)
 6:   Set2 ← SelectAllRightPointsFromPos(BDPoints, delta)
 7:   SubSets ← Append(SubSets, Append(Set1, Set2))
 8:   BDPoints ← RemoveSelectedPoints(BDPoints)
 9:   Return SegmentBifurcationDiagram(BDPoints, SubSets, delta)
10: end if
```

Figure 3.6 (a) shows an example where the distance *delta* is not taken into consideration to rule out segment sections that do not belong together. In this example we took all the points observed from below starting from the lowest point. As we can observe in Figure 3.6, if the algorithm takes all the points that can be seen from below starting from the lowest point, the algorithm will take all the blue points. Notice that not all the blue points correspond to a single segment. The blue points where $r \in [0, 12]$ correspond to another segment. Figure 3.6 (b) shows the correct segmentation by using the parameter *delta*.

The way quantitative data is generated does not guarantee that the points are continuous; there may be values of r for which not all fixed points of the system were

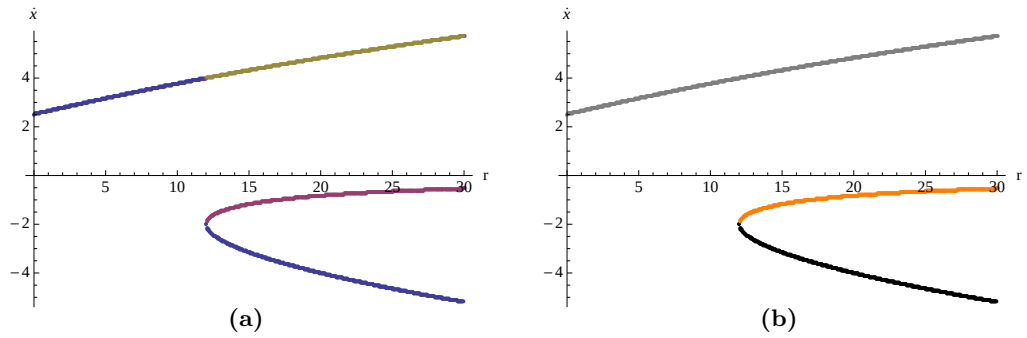


Figure 3.6: Bifurcation diagram segmentation process. (a) In the bifurcation diagram all the blue points corresponds to the points that can be observed from below (blue points). But not all the blue points belong to the segment. (b) The segmentation algorithm consider the *delta* parameter to discriminate points.

determined. That produces discontinuous streams of data, where for those indeterminate values of r a data item are taken from a different segment; therefore, it is necessary to check distances between neighboring points, and discard points that are too far away (*delta*). When we are filtering the last layer of points, and no x was determined for a value of r, no data is inserted in that place, creating holes in the segments. In either case, a segment with discontinuities or holes has to be recognized and reassembled back together to form a single BDS.

Figure 3.7 shows an example of this case illustrating gaps in the bifurcation diagram. At r^* there is a hole in the black segment. If we scan to determine the point seen from below, we will take the orange point; this point do not belong to the black segment (because of the hole). We use *delta* to verify if a point belong or not to a segment. If we do not check the distance between neighbors we could include those points in the wrong segment.

After selecting the points at the left, we continue with the same process to the right (selecting all the points that can be seen from below to the right of the lowest point). Both sets (left and right) are placed in a single segment.

Figure 3.8 shows the BD for a system defined by the differential equation $\dot{x} = rx + x^3$. The first step is to locate to the lowest point of the BD. From that point we take all the points that can be seen from below to the left and then to the right. In this example

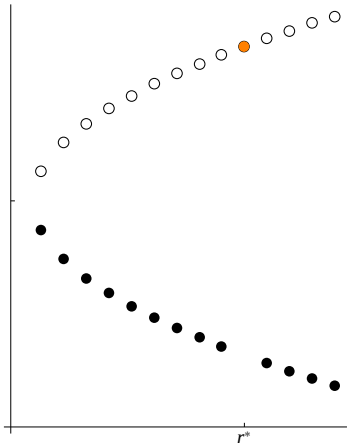


Figure 3.7: Gaps in a bifurcation diagram.

(Figure 3.8) we observe that from the lowest point there are no points to the left, but there are points that can be seen to the right of the minimum. We take those points and remove them from the BD (Figure 3.9) and continue until no points remain. Figure 3.10 shows the diagram right before the last step.

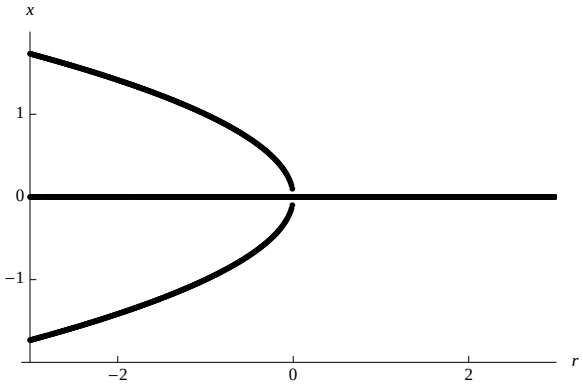


Figure 3.8: Bifurcation Diagram defined by $\dot{x} = rx + x^3$.

3.4.3. Monotonic Segments

Given the subsets of points obtained in the first step, a qualitative derivative is determined for each point contained in each subset (Line 5). For this function W is the window size of the kernel function (further information about the kernel function is described

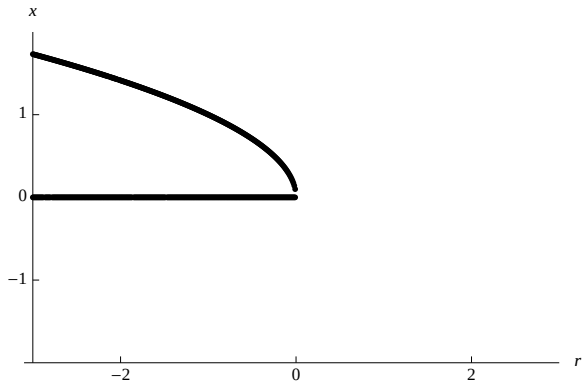


Figure 3.9: Removing the points that can be seen from below of the example in Figure 3.8.

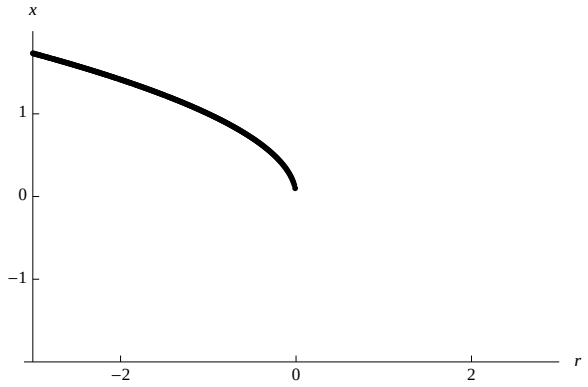


Figure 3.10: Continue removing the points until no points remain of the example in Figure 3.8. This Figure illustrate right before the last step.

in Section 3.5). Line 6 detects the positions where exists a change in the qualitative slope (i.e., change from a positive slope to a negative one); these changes define the initial and final points of the MS. The subsets are segmented into regions having the same qualitative behavior (Line 7), taking as breaking points the position changes defined before; each such region is called a Monotonic Segment (MS). To obtain the qualitative slope of each point, we use a qualitative noise filter developed by Kay et al [Kay00]. This filter produces a qualitative derivative for each point of the subset; it operates by applying a qualitative kernel function to a window of fixed size (W), that is slid across the subset. Algorithm 3 defines the complete algorithm of the process of segmenting the information into MSs.

The kernel function defined in Section 3.5 determines the slopes of all the points contained in a subset (positive, negative, or unknown). Given all the slopes, we traverse that subset searching for the points where the slope or stability changes from one point to the next. We take all the points found by the search into new subsets before finding any points with different slopes or stabilities. This action is repeated until no points remain. As noted above, all the points contained in each new subset have the same slope and stability and constitute a MS. An MS is represented by its beginning and end, its nature (stable or unstable), and its qualitative behavior (positive, zero, or negative slope).

Algorithm 3 `BreakingInToMS`$(Subsets, MSs, W)$

```
1: if Subsets is Empty then
2:    Return MS
3: else
4:    TemporalSet ← First(Subsets)
5:    Slopes ← ApplyQualitativeFilter(TemporalSet, W)
6:    PosChanges ← FindChangesInSlope(Slopes)
7:    TemporalMS ← BrakeSubsetFromPosChanges(TemporalSet, PosChanges)
8:    Return BreakingInToMS(Rest(Subsets), Append(MS, TemporalMS), W)
9: end if
```

If we take the example of Figure 3.8. In fist step, we brake the BD in sets where all the points belong to a function. We use the qualitative filter defined in section 3.5 to define the qualitative derivatives of each point. We make a selection to form the monotonic segments depending on its behavior (slope) and its nature (stability). Figure 3.11 shows an example of a subset that has been broken into two MSs. One MS has a positive slope

and the other one has a zero slope.

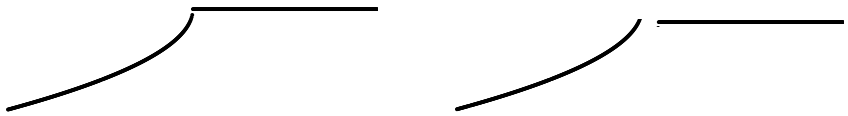


Figure 3.11: The points of a subset (a) before and (b) after being broken into two MSs.

3.4.4. Gluing Segments

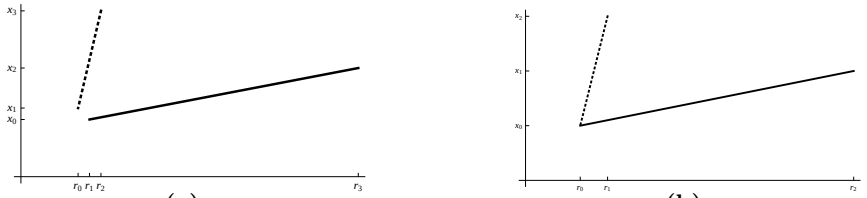
In the previous steps the algorithm works with quantitative data. In this step the system works with the initial and final points of each MS (at a qualitative level). The monotonic segments produced by the previous step may leave gaps between the segments due to imprecision and coarse granularity of the initial quantitative information. The bifurcation diagram is to be continuous; i.e., there must be no gaps between segments.

In addressing the gaps issue, two cases can be distinguished. The first case deals with scenario where several adjacent segments separated by holes genuinely form a single segment and they must be glued into a single one. In this first case a sequence of segments that belong together are glued to form a single segment. These segments must be close enough to be considered part of the same segment. If both segments have the same qualitative behavior, they are merged into one. The algorithm search an adjacent segment (Line 2) and the resulting adjacent MS is aproximante to the current segment (Line 3). Figure 3.12 shows an example of the holes produced by the qualitativization of a segment of a BD and the result of merging several segments when the segments have the same qualitative behavior.

The second case occurs when the extremes of a pair of segments are very close, but their natures or slopes are not the same. In this case, each segment is extended until the segments meet, maintaining their slope and nature. Figure 3.13 shows an example of this case. Algorithm 4 shows the complete process of gluing segments.

(a) (b)

Figure 3.12: BD with gaps or holes (a) before and (b) after gluing the segments



(a) (b)

Figure 3.13: Two nearby segments that are not to be merged into one due to their different qualitative nature – (a) before and (b) after filtering

3.4.5. Setting Landmarks

After gluing segments together to reestablish continuity, we generate the qualitative representation defined in Chapter 3.4. Each end point of an MS has associated r and x values. These values are given symbolic names, called landmarks, and are stored in the corresponding variable's *quantity space* in order of their quantitative magnitudes. The setting landmarks process start by taking the initial and final points of the MSs (Line 3 and 4).

A symbolic name is associate to this value (Line 5) and then the symbolic name (landmark) is inserted in QS (Line 6). Note that at the time of insertion of the new landmark, it has to be at the right position. Algorithm 5 shows the complete process of setting the landmarks.

A quantity space is a set of landmarks ordered by magnitude, i.e., the value of

Algorithm 4 `GluingSegment`$(MSs, DeltaGlue)$

1: **for** `All` MS `in` MSs **do**
2: $AdjacentMS \leftarrow$ `SearchAdjacentMS`$(MS, MSs, DeltaGlue)$
3: $MS \leftarrow$ `ApproximateMS`$(MS, MSs, DeltaGlue)$
4: **end for**
5: `Return` MSs

the i_j landmark is greater (in magnitude) than the i_{j-1} landmark. The real quantitative value of each landmark may or may not be known; that value is discarded anyway, since the system only needs an order relation between landmarks. An example of a quantity space is $QSx = \{x_0, x_1, ..., x_n\}$, where $x_0 < x_1 ... < x_n$.

During the simulation or planning process the real landmark values are not used. All we know is an order relation. During the planning algorithm, new landmarks are generated, whose real values are not known. It is only known that the new landmark is between two landmarks

The Quantitative Space (QS) is the set of landmarks ordered by magnitude. The real values of the landmarks are discarded, but to create the set of landmarks, initially their real values must be known. Before eliminating any quantitative value, the system stores all the numerical values (left or right extremes of a segment). The r values are separated from the x values and ordered by magnitude. For each element or value contained in both sets, the algorithm substitutes the numerical value by a qualitative one (i.e., 0.2 by x_1, 0.05 by x_2, ...). The obtained sets will be considered as the QS of the qualitative representation of the system.

Algorithm 5 `SettingLandmarks`(MSs)

1: $QS \leftarrow \{\}$
2: **for** `All` MS `Contained in` MSs **do**
3: $PointLeft \leftarrow$ `TakeInitialLeftPoint`(MS)
4: $PointRight \leftarrow$ `TakeInitialRightPoint`(MS)
5: $Landmark \leftarrow$ `AssociateSymbolicName`$(PointLeft)$
6: $QS \leftarrow$ `AppendOrderedByMagnitude`$(QS, Lendmark)$
7: $Landmark \leftarrow$ `AssociateSymbolicName`$(PointRight)$
8: $QS \leftarrow$ `AppendOrderedByMagnitude`$(QS, Lendmark)$
9: **end for**
10: `Return` QS

3.4.6. Transitions

To analyze or simulate a dynamical system, means to determine how the system will behave when a parameter r varies. We need to specify the changes r undergoes. Since we are not interested in the precise value of time when the events happen, that description

will not include time explicitly, it only includes the order in which they happen. During the qualitative simulation, we want to know what MSs the system will traverse and the order of the transition events, given changes in the value of a parameter or given a perturbation to the state variable.

The qualitative representation stores each of the four transition directions. In Line 1UpT stores the transitions that may occur in the presence of a positive perturbation; In Line 2$DownT$ does the same for a negative one. If the parameter is incremented and the system goes beyond the segment, the segment the system will transit to is stored in *Right* (Line 3). Finally, if the parameter is decremented and the system goes beyond the segment, the system stores the transition in $Left$ (Line 4). Algorithm 6 shows the process of defining the Transitions of the MSs.

Algorithm 6 `GenerateTransitions`(MSs)

1: $UpT \leftarrow$ `PerturbationTransitions`(MSs, Up)
2: $DownT \leftarrow$ `PerturbationTransitions`($MSs, Down$)
3: $LeftT \leftarrow$ `IncDecTransitions`($MSs, Left$)
4: $RightT \leftarrow$ `IncDecTransitions`($MSs, Right$)
5: `Return` ($UpT, DownT, LeftT, RightT$)

When a perturbation occurs (Algorithm 7), if the system is at a stable fixed point, the system will return to the same stable fixed point (Line 6), the boundary of the perturbation is the same as the segment's boundary. The algorithm records in $PertTrans$ and continue with the next MS. If the system is at an unstable fixed point, a positive perturbation occurs, and there exists a stable fixed point above it, then the system will *transition* to that fixed point. If there is no such stable fixed point above, the system will go to infinity (i.e., it *blows up*). A similar behavior occurs for negative perturbations, with an unstable system moving to a stable point below, or to minus infinity.

The transition segment of a perturbation depends where the perturbation took place. Therefore the algorithm defines all possible transitions in case of a perturbation across a segment. The Perturbation algorithm defines the limits of the search in Line 3 and 4; $InitialPoint$, for the left end of the segment, and $EndPoint$ for the right end. The algorithm searches the closest stable segment in the vertical direction (Line 10) from

InitialPoint. The boundary of the perturbation (Algorithm 8) is defined considering the limits of *PertTemp* and other segments that may cross between MS and *PertTemp* in Line 12. The boundary defines the limits where the perturbation may occur. In Line 14, the algorithm updates $InitialPoint \leftarrow Limit$ and continues the search of another region of perturbation until *InitialPoint* reaches *EndPoint* (Line 9). The Perturbation Transition process is defined in Algorithm 7.

Algorithm 7 `PerturbationTransitions`$(MSs, Direction)$

```
1:  PertTrans ← {}
2:  for All MS in MSs do
3:    InitialPoint ← TakeInitialPoint(MS)
4:    EndPoint ← TakeEndPoint(MS)
5:    if MS is stable then
6:      Pert ← (MS, InitialPoint, EndPoint)
7:    else
8:      Pert ← {}
9:      while InitialPoint < EndPoint do
10:       PertTemp ← ClosestVerticalMS(MS, InitialPoint, MSs)
11:       if PertTemp is not Empty then
12:         Limit ← DefineBoundary(MSs, PertTemp, InitialPoint, EndPoint)
13:         Pert ← Append(Pert, (PertTemp, (InitialPoint, Limit)))
14:         InitialPoint ← Limit
15:       end if
16:     end while
17:   end if
18:   PertTrans ← Append(PertTrans, Pert)
19: end for
20: Return (PertTrans)
```

When there is an increment or decrement in the parameter. Two types of events may occur. One if there is an adjacent segment to MS. The other type of event that may occur is what we call a *fallOff*. When the system is at a stable fixed point and parameter r increases or decreases, the system will travel along the segment to which the fixed point belongs. If the segment ends before the change in the parameter has completed, then there is no other fixed point along that segment i.e., at that state. The system will transition to the nearest attracting fixed point or to infinity. A more detailed explanation of these events can be found in the work of Flores et al [Flores06].

Algorithm 8 `DefineBoundary`$(MSs, PertTemp, InitialPoint, EndPoint)$

```
1: PertEnd ← TakeEndPoint(PertTemp)
2: if PertEnd < EndPoint then
3:    Limit ← PertEnd
4: else
5:    Limit ← EndPoint
6: end if
7: Boundary ← SearchLowerMS(PertTemp, MS, MSs, PertTemp, Limit)
8: if Boundary is Empty then
9:    Return (Limit)
10: else
11:   Return (Boundary)
12: end if
```

Algorithm 9 defines the case of an increment or decrement in the parameter process. Algorithm *IncDecTransitions* takes the end of the segment according the search direction in Line 3(Limit). i.e., left end for the Left transition and right end for the Right transition. In Line 4 the algorithm searches and adjacent segment by the function *SearchAdjacentSegment* and store it in *IncDecTrans* (Line 5). This process is repeated for all MS in MSs.

Algorithm 9 `IncDecTransitions`$(MSs, Direction)$

```
1: IncDecTrans ← {}
2: for All MS in MSs do
3:    Limit ← TakeEndSegment(MS, Direction)
4:    TransitionSegment ← SearchAdjacentSegment(MS, MSs, Direction)
5:    IncDecTrans ← Append(IncDecTrans, TransitionSegment)
6: end for
7: Return (IncDecTrans)
```

Figure 3.14 shows an example of an artificial QBD used to illustrate the generation of possible transitions in a BD. In the diagram, we assume the segment S_0 is unstable. Lets consider two different situations by which the system will move to a fixed point that does not belong to the same segment. If a positive perturbation occurs between the landmarks r_0 and r_3, the system will go to the fixed point above the point where the perturbation occurs; i.e., segment S_1. If the system is at a stable fixed point in segment S_1, and parameter r grows from r_0 towards r_4, the system will remain at segment S_1 until reaching r_3. Since

segment S_1 ends at that point, the system will *fall-off* to segment S_3 at r_3. Table 3.3 shows all possible transitions for BD of Figure 3.14.

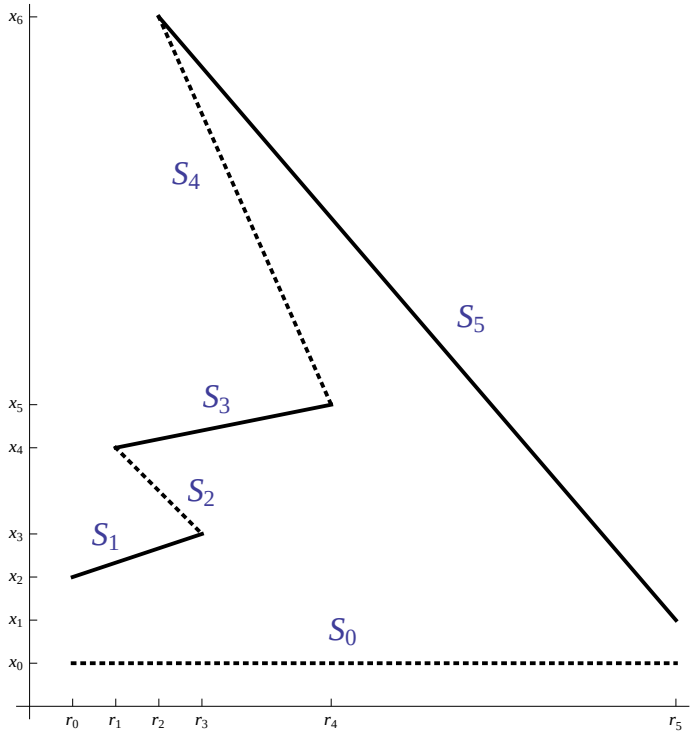


Figure 3.14: Artificial Qualitative Bifurcation Diagram used to Illustrate the Representation of Transitions (stable and unstable segments are represented by solid and dashed lines, respectively)

Table 3.3: Transitions of the Bifurcation Diagram on Figure 3.14.

Segment	Up	Down	Left	Right
S_0	$\{S_1,\{r_0,r_3\}\}$ $\{S_3,\{r_3,r_4\}\}$ $\{S_5,\{r_4,r_5\}\}$	{}	{}	{}
S_1	$\{S_1,\{r_0,r_3\}\}$	$\{S_1,\{r_0,r_3\}\}$	{}	S_3
S_2	$\{S_3,\{r_1,r_3\}\}$	$\{S_1,\{r_1,r_3\}\}$	{}	{}
S_3	$\{S_3,\{r_1,r_4\}\}$	$\{S_3,\{r_1,r_4\}\}$	S_1	S_5
S_4	$\{S_5,\{r_2,r_4\}\}$	$\{S_3,\{r_2,r_4\}\}$	{}	{}
S_5	$\{S_5,\{r_0,r_5\}\}$	$\{S_5,\{r_0,r_5\}\}$	S_3	{}

3.5. Qualitative Filter

The qualitative filter is defined in the work of Kay in Semi-Qualitative System Identification [Kay00]. The qualitative filter brakes a set of points into monotonic subsets, where all the points in a monotonic subset have the same derivative sign (+ for positive, - for negative). Each sign is obtained using a qualitative kernel function (Equation 3.1). This function defines the slope of a particular point using its neighbor points. For certain cases the the kernel function cannot distinguish when derivative is the zero. For these cases it is necessary to obtain the standard deviation of the points (Equation (3.3)).

The kernel function is applied to a window of size n; this window is slid across the set of points. For every application of the kernel function we obtain a qualitative derivative. + if the the points of the window are monotonically increasing, - if the points are monotonically decreasing, and * if its slope can not be defined.

The kernel function defines the slope of a point using a window of points. This function is defined by Equation (3.1). Where i is i-th term in the window. The terms $\bar{t}$ and $\bar{y}$ are used for the average of each variable of the whole set. The result of Equation (3.1) is taken in a qualitative way. Form result of Equation (3.1) we obtain the qualitative derivative in Equation (3.2)

$$slope_j = \left(\frac{\sum_{i=1}^{n} (t_i - \bar{t})(y_i - \bar{y})}{\sum_{i=1}^{n} (t_i - \bar{t})^2} \right) \tag{3.1}$$

$$qslope_j = Sign(slope_j) \tag{3.2}$$

Since the information may have numeric imprecision, noise, etc., it is necessary to obtain the standard deviation of the points. For this case we use Equation (3.3), where σ_v is the standard deviation of the points. Therefore, if $|slope| \leq 3.5s$ the $qslope$ is unknown (i.e., *). The sign returned by the kernel function is based on a 3.5 σ confidence range, this gives a 99 % certainty that the slope is not zero [Kay00].

The qualitative filter is applied to all the fixed points, and detects the changes in directions. This mean that where a change in the qualitative slope occurs, the algorithm

takes all the points with the same slope, and continues evaluating. Note that, when a change in direction (slope) occurs (i.e., going form positive to negative slope) there is a point with a zero slope. For our particular representation this point is not defined in it. For this case, the qualitative representation takes only two segments. For example if we observe Figure 3.15, we notice that at the beginning the first points have a positive slope, and later a change in direction occurs. The following points change their qualitative slope. During the change, exists a point with zero slope. The change occurs instantly, therefore this particular point (a point with zero slope) is not considered in the representation.

$$s_j = \frac{\sigma_v}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2}} \tag{3.3}$$

As an example, consider the set of points plotted in Figure 3.15. In the figure we can observe an increasing trend on the first part of the set, and a decreasing trend on the second part of them. If we get the derivative with the traditional all across the set of points. We get in some part of the increasing trend, that there are some really small part that has the opposite slope (negative); we will be capturing the noise. When we use the qualitative filter to determine a qualitative slope for a particular point, we define it using the next n (window size) points of the set. That window of points give us more information to define the qualitative slope. The result of applying the qualitative filter to Figure 3.15 can be observed in Figure 3.16. Figure 3.16 shows that the qualitative filter is focus in determine chances in derivatives of the data.

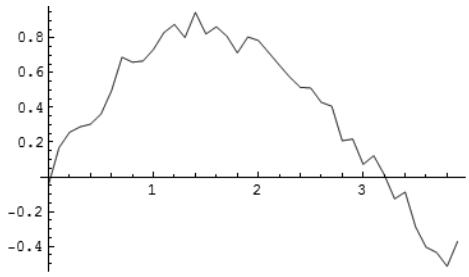


Figure 3.15: An example of a set of point with noise

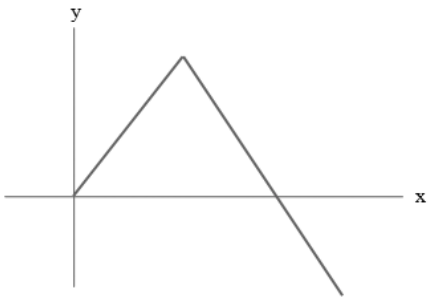


Figure 3.16: After applying the qualitative filter in the example defined in the Figure 3.15

3.6. An Example of Qualitative Generation

Consider the system defined by the differential equation $\dot{x} = rx + x^3$. Its bifurcation diagram is generated using Flores' application [Flores10], and the result is shown in Figure 3.17 (a). From the quantitative information the algorithm splits the diagram in sets where all the points contained in each set is defined by a function (Figure 3.17 (b)). In the Figure 3.17 (b) each set is defined by a different color.

The sets are used to define the monotonic segments using the qualitative filter. These segments have the same nature and behavior. These mean that every point contained in each MSs has the same slope and the same stability. That is why, we take only the initial and final points of each MS and draw a line to join them. This representation can be observed in Figure 3.17 (c). The black lines correspond to stable segments and the red lines belong to unstable segments. As we can observe in Figure 3.17 (c) there are small gaps in the diagram. In order to preserve continuity we join the segments according the to rules explained above.

Finally, we take that information an translate it to the qualitative representation defined in Chapter 3. Table 3.4 shows the qualitative representation of the bifurcation diagram. Figure 3.18 shows the qualitative diagram of the BD of the Figure 3.17 (a).

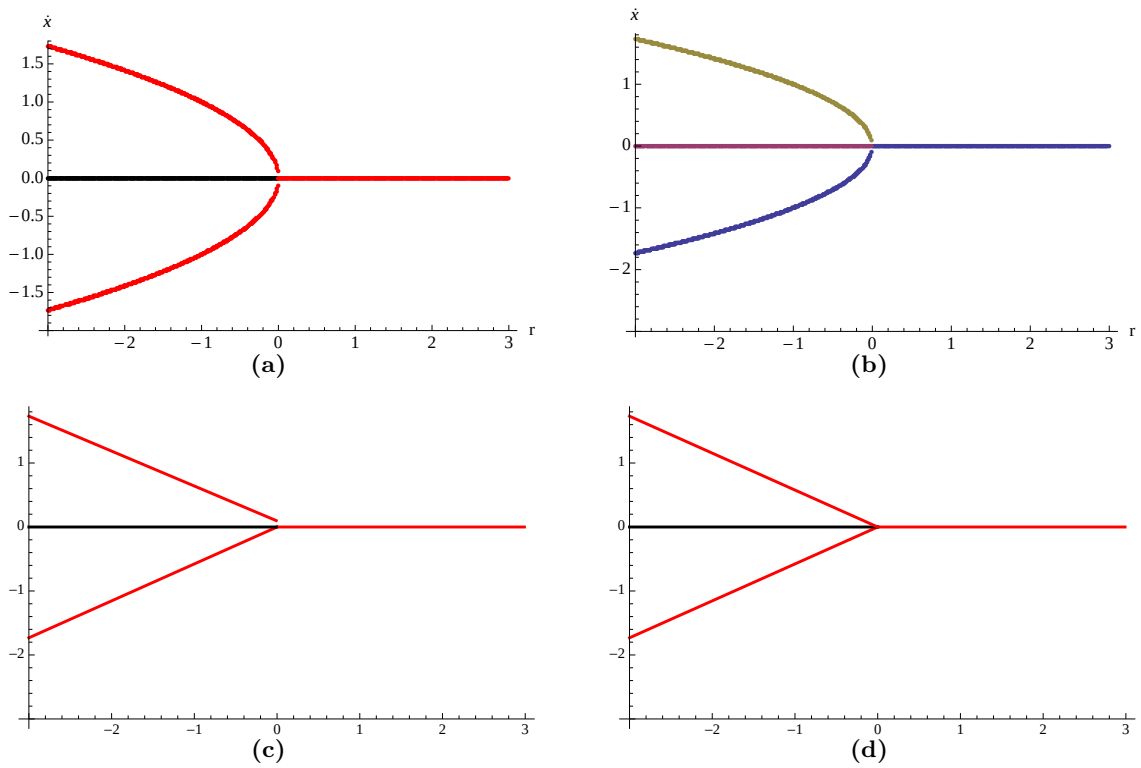


Figure 3.17: The progress of getting the essential characteristics of a bifurcation Diagram defined by the differential equation $\dot{x} = rx + x^3$. (a) The algorithm starts from the bifurcation points (b) it brakes these points in sets where all the points are defined by a function, (c) brake the function sets in MSs, (d) An continuity is reestablished.

Table 3.4: Qualitative Representation of the Bifurcation Diagram of Figure 3.18.

Variables		$\{r, x\}$
Quantity Space	QS_r	$\{r_0, r_1, r_2\}$
	QS_x	$\{x_0, x_1, x_2\}$
BDS	S_1	$\{\{r_0, x_0\}, \{r_1, x_1\}, \{+, u\}\}$
	S_2	$\{\{r_1, x_1\}, \{r_2, x_1\}, \{0, s\}\}$
	S_3	$\{\{r_0, x_1\}, \{r_1, x_1\}, \{0, u\}\}$
	S_4	$\{\{r_0, x_2\}, \{r_1, x_1\}, \{-, u\}\}$
Transitions		See Table 3.5

3.7. Chapter Conclusions

As mentioned in this chapter there is no single, unique, or best qualitative representation; there may be more than one of them . For our particular problem of predicting the

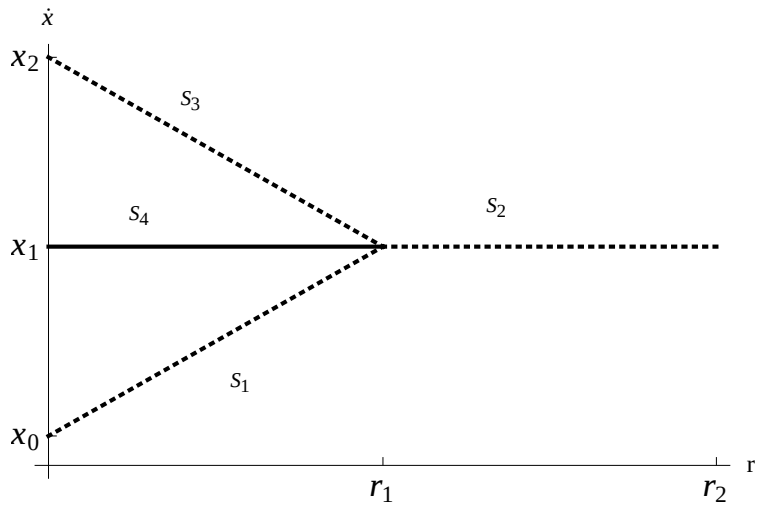


Figure 3.18: Qualitative Bifurcation Diagram defined by the differential equation $\dot{x} = rx + x^3$.

Table 3.5: Transitions of the Bifurcation Diagram on Figure 3.18.

Segment	Up	Down	Left	Right
S_1	$\{S_4,\{r_0, r_1\}\}$	$\{\}$	$\{\}$	$\{\}$
S_2	$\{\}$	$\{\}$	$\{S_4\}$	$\{\}$
S_3	$\{\}$	$\{S_4,\{r_0, r_1\}\}$	$\{\}$	$\{\}$
S_4	$\{S_4,\{r_0, r_1\}\}$	$\{S_4,\{r_0, r_1\}\}$	$\{\}$	$\{S_2\}$

behavior of a system represented by a bifurcation diagram, we need to capture the essential characteristics of the system and translate them into an optimal qualitative representation. In this chapter we proposed a qualitative representation which includes as a tuple of four elements (variables, quantitative spaces, segments, and transitions).

Also, in this chapter we presented an algorithm that captures the essential characteristics of a bifurcation diagram, and translate them into the qualitative representation proposed . The algorithm starts from the quantitative information of the BD (bifurcation points), breaks the diagram in sets where all the points form a function. Those sets are converted to monotonic segments (segments with same behavior and nature). We use the initial and final points of the monotonic segment to create the qualitative segments. With the qualitative segments defined, it is possible to define the transitions of every segment.

Simulation and planning algorithms will use the qualitative representation of the

bifurcation diagram. These algorithms will use only the information provided by the qualitative representation generated by this algorithm.

Chapter 4

Qualitative Simulation and Control Planning

A dynamical system is a physical system that evolves with time. It can be modeled by a set of differential equations. From the differential equations we can obtain a bifurcation diagram. The BD can be used as an analysis tool, and it describes the stable operational limits of the dynamical system.

Based on flow concepts to predict the evolution of a dynamical system, and using only qualitative derivatives, books like Non Linear Dynamics and Chaos by Strogatz [Strogatz94] describe the dynamic of the system. Using those principles we developed two algorithms, one (simulation algorithm) to predict the behavior of the dynamical system when one of its parameters is varied. The second one (planning algorithm) describes the parameters changes that the system needs to transition from an initial state to a final state.

In this chapter the simulation and planning algorithm are presented. Both algorithms start from the qualitative representation (Chapter 3) of the bifurcation diagram and their output is in purely qualitative terms.

4.1. Qualitative Simulation

Given a qualitative representation of a bifurcation diagram, we can simulate the behavior of the represented system in response to perturbations and changes in its parameter r. Given an initial state x of the system, located at some point on a segment of the bifurcation diagram, we can trace how the system will move along the bifurcation diagram as r moves from landmark to landmark or to a point between existing landmarks. We can also simulate the effects of perturbations to the system.

A dynamics descriptor is a list of actions indicating successive values of r that the system parameter will attain, plus any perturbations and their directions (either +, or -), made to the system. The pseudo-code for performing Qualitative Simulation is shown in Algorithm 10.

The input to QSimulation (Algorithm 10)is a qualitative bifurcation diagram Q, a state in Q $\{S_{current}, r_{current}\}$ (state is defined as a position in a segment), a dynamics descriptor (D), and the history of movements of the simulation (H). Changes in D are stored in the form of a landmark if the system goes from the current state to the position that indicates the landmark. A parameter change in D is represented as a landmark value to which the parameter is to be adjusted. For positive or negative perturbations I use the symbols + and - respectively. Initially, the history (H) is empty; the simulation is performed by successively removing the first element of D, until it becomes empty. If the removed element is a perturbation, function $NextState$ determines the next segment the system will reach (Line 6). If the current segment $S_{current}$ is unstable, the next segment is determined by consulting the transition table, T, in Q. If $S_{current}$ is stable, the system will remain in the same segment. In either case, the parameter value r is unchanged. The algorithm updates the segment, appends the perturbation change to the history ($\texttt{Append}(H, \texttt{Perturbation}(S_{current}, S_{new}, r_{current}, a))$) in Line 10, and continues with the next action indicated by the descriptor D (Line 11).

If the element removed from D indicates a new value a for the parameter r, there is an increase or decrease in r. In the case that the current segment $S_{current}$ is not stable with a slope different from zero, then a $Derailment$ will occur (Line 14). The system will

Algorithm 10 QSimulation$(Q, \{S_{current}, r_{current}\}, D, H)$

```
1: if D is Empty then
2:   Return H
3: else
4:   a ← TakeFirst(D)
5:   if a is a Perturbation then
6:     S_new ← NextState({S_current, r_current}, a, Q)
7:     if S_new is Empty then
8:       Return Append(H, BlowUP(S_current, S_current, Undefined))
9:     end if
10:    H ← Append(H, Perturbation(S_current, S_new, r_current, a))
11:    Return QSimulation(Q, {S_new, r_current}, D, H)
12:  else
13:    // If D is not a perturbation, it is a change in r
14:    if S_current is Unstable and Slope of S_current ≠ 0 then
15:      //A Derailment occurs
16:      S_new ← NextState({S_current, r_current}, a, Q)
17:      if S_new is Empty then
18:        Return Append(H, BlowUP(S_current, S_current, Undefined))
19:      end if
20:      x_behave ← DefineBehaveX(S_current, S_new)
21:      r_behave ← DefineBehaveR(r_current, a)
22:      H ← Append(H, Derailment(S_current, S_new, r_current, x_behave))
23:      D ← InsertAtFirstPosition(D, a)
24:      Return QSimulation(Q, {S_new, r_current}, D, H)
25:    else
26:      if a ϵ S_current then
27:        r_new ← a
28:        H ← Append(H, Follow(S_current, r_current, r_new, , r_behave, x_behave))
29:        Return QSimulation(Q, {S_current, r_new}, D, H)
30:      else
31:        // The system moves to an adjacent segment
32:        {S_new, r_new} ← NextState({S_current, r_current}, a, Q)
33:        if S_new is Empty then
34:          Return Append(H, BlowUP(S_current, S_current, Undefined))
35:        end if
36:        D ← InsertAtFirstPosition(D, a)
37:        x_behave ← DefineBehaveX(S_current, S_new)
38:        r_behave ← DefineBehaveR(r_current, a)
39:        H ← Append(H, Follow(S_current, r_current, r_new, r_behave, x_behave))
40:        if a Falloff Occurs then
41:          H ← Append(H, FallOff(S_current, S_new, r_current, x_behave))
42:        end if
43:        Return QSimulation(Q, {S_new, r_new}, D, H)
44:      end if
45:    end if
46:  end if
47: end if
```

Algorithm 11 `NextState`$(\{S_{current}, r_{current}\}, a, Q)$

```
1: Column ← DefineTransitionColumn(S_current, r_current, a)
2: T ← TakeTransitions(Q)
3: Row ← TakeRow(S_current)
4: if a is a Perturbation then
5:    S_next ← TakePerturbationTransition(Row, Column, r_current)
6:    Return S_next
7: else
8:    {S_next, r_next} ← TakeIncDecTransition(Row, Column, Q)
9:    Return {S_next, r_next}
10: end if
```

move to another fixed point, at a segment defined by the function $NextState$ (Line 16). It then reinserts action a at the front of D (Line 23), so that we try to reach a again, since we moved to another segment before reaching a. The derailment that occurred is added to the history in Line 22, before considering the next element of D. If a describes a point that is within the bounds of the current segment ($a \epsilon S_{current}$) which is a stable segment or segment with slope zero (Line 26), the value of parameter r will change to a ($r_{new} = a$) in Line 27, within segment $S_{current}$, with the value of x changing according to the direction of change of r and the slope of $S_{current}$.

In the remaining case, if the new value a of r is not within the extent of segment $S_{current}$, the system state will change to another segment. Upon reaching the end of the current segment, the system state will transition to another segment; function $NextState$ in Line 27 determines the new segment using the transition table T. Current state is updated to (S_{new}, r_{new}), and this transition is added to the history in Line 39. If there is no neighboring segment in that direction, a *fallOff* occurs; the system adds the *fallOff* to the history. The system inserts the action "move to a" at the first position of D, because parameter r has not yet reached a (Line 36). We repeat the same process, either remaining within the segment, if a is within the reach of the segment, or falling off again, until eventually r reaches a.

When D becomes empty, the algorithm returns the history H (Line 2), recorded during the simulation. History H represents what occurs in the system from applying the descriptor D. In the history, we use $Follow$ to indicate that the parameter continues along a given segment from one position to another, $Perturbation$ to indicate that a perturbation

event occurs at a defined position, *fallOff* to indicate that the system has reached the end of the MS it was at, and it is transitioning to another MS in the same direction, and *Derailment* to indicate that a derailment occurs in the system, moving from one segment to another at a defined position. We use r_{behave} and x_{behave} to specify if the variable x or the parameter r is incremented, decremented o remain in the same qualitative state. Table 4.1 shows the set of events considered in simulation by QBD.

Notice that the function $NextState$ Algorithm 11 returns the next state in case of a perturbation or an increment or decrement in the parameter. This function receives as an input the current state, the action (a) i.e., perturbation, and the qualitative representation (Q); this function returns the next segment if the action is a perturbation in Line 5 and return the next state if the action is an increment/decrement of the parameter (Line 8). The next state process is defined in Algorithm 11.

Table 4.1: List of events that can be recorded in the history H

Event	Description
Follow$[S_{current}, r_{initial}, r_{final}, r_{behave}, x_{behave}]$	The system continues along the segment from $r_{initial}$ to r_{final}
FallOff$[S_{current}, S_{final}, r_{current}, x_{behave}]$	Fall off from $S_{current}$ to S_{final} at $r_{current}$
Derailment$[S_{current}, S_{final}, r_{current}, x_{behave}]$	Derailment occurs from $S_{current}$ to S_{final} at $r_{current}$
Perturbation$[S_{current}, S_{final}, r_{current}, r_{behave}]$	Perturbation occurs at $r_{current}$, take the system from $S_{current}$ to S_{final}
BlowUp$[S_{current}, r_{current}, Undefined]$	The system goes to an undefined state

4.1.1. Simulation Example

To illustrate the behavior of the simulation algorithm let us use the qualitative bifurcation diagram described in Chapter 3, Figure 3.4. As mentioned above, the algorithm receives as input the qualitative representation of the BD, as defined in Table 3.1, an initial state in the diagram, and the changes to the parameter r ($Descriptor$) defined in D. For this example let us consider the system starts in segment S_0 at r_3, and the parameter changes are $D = \{r_4, +, r_3\}$.

The first action is taking the first element in D (r_4). This means that the system

needs to move from r_3 to r_4. As we can observe in Figure 4.1, r_4 does not belong to segment S_0. Since r_4 is lower in magnitude than r_3, the system moves across the segment until its left end. Now, it is necessary to find the next segment ($NextState$) that the system will reach once the system leaves the actual segment (S_0). The function $NextState$ uses the Transition Table (Table 3.2) to define which segment the system will reach. The Transition Table defines that the system will reach segment S_2 and a *FallOff* will occur at r_1. The landmark r_4 is inserted in D because the system has not arrived yet to the partial destination r_4. This part of the history is recorded in H.

At this point the system is located in segment S_2 at r_1. Once again the algorithm takes the first element in D (r_4). Since a point at r_4 belongs to segment S_2, the system moves across the segment until reaching landmark r_4. The algorithm updates the current point to r_4 and records the history in H.

At this point the system is located in segment S_2 at r_4 and the descriptor is defined as $D = \{+, r_3\}$. The algorithm takes the first element (+ a positive perturbation occurs in the system). Since the system is located in a stable segment, the system returns to the same point after the perturbation. The perturbation is recorded in H and the algorithm continues taking the first element of D until no elements remain.

Figure 4.1 shows graphically the result of simulation process for this example. The QBD corresponds to the system defined by the differential equation defined by $\dot{x} = 16 + rx - x^3$. Dotted and continuous lines correspond to unstable and stable segments respectively, and the blue lines to the path that the simulation follows. Table 4.2 shows the output (history) of the simulation algorithm. The history of the simulation describes step by step the behavior of the system when one of its parameters (r) is varied.

4.2. Semi-stable Fixed Points

This section revisits the discussion about semi-stable fixed points started in Section 3.3. For instance, in the previous example, if the system started at segment S_0, with r decreasing, when the system reached the left end of S_0 it undergoes a derailment, ending up in segment S_2. Now, let us assume point p_1 is included in the representation. The system

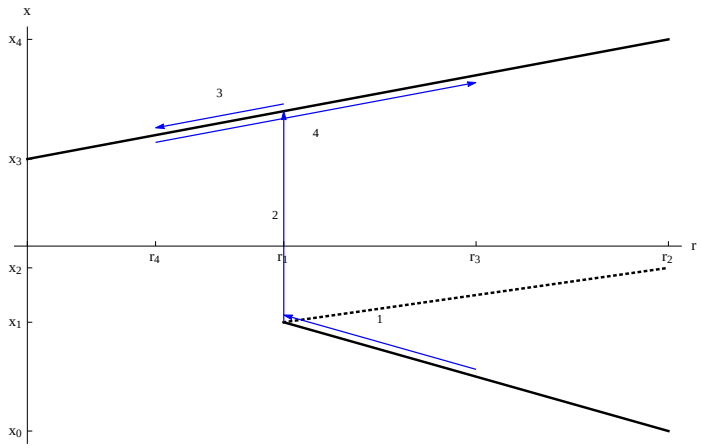


Figure 4.1: Simulation example in the bifurcation diagram of the system defined by $\dot{x} = 16+rx-x^3$. The continuous lines correspond to a stable segment, the dotted lines correspond to unstable segments, and the blue lines to the path that the system follows according to the simulation algorithm.

Table 4.2: The output after applying the simulation algorithm to example described in Figure 3.4

	Action	Description
1	$Follow[S_0, r_3, r_1, -, +]$	The system follows from r_3 to r_1 in the segment S_0, r is decremented (-), and x is incremented (+).
2	$FallOff[S_0, S_2, r_1, +]$	The system falloffs from S_0 to S_2 at the position r_1.
3	$Follow[S_2, r_1, r_4, -, -]$	The system follows from r_1 to r_4 in segment S_2, r is incremented (+).
4	$Perturbation[S_2, S_2, r_4, 0]$	The system occurs a perturbation, since the segment S_2 is stable, it returns to the S_2 at r_4.
5	$Follow[S_2, r_4, r_3, +, +]$	The system follows from r_4 to r_3 in the segment S_2. r is incremented (+), and x is incremented.

transitions to point p_1 and instantaneously after, it undergoes a derailment, jumping to segment S_2 (Figure 4.2).

The derived behavior of the system does not change with the inclusion of the semi-stable point (and its corresponding segment), and the simulation output would include an unnecessary transition to that region. Given the above, we decided not to include semi-stable points in the representation.

Figure 4.2 shows the behavior of the system when it stars from a point in the segment S_0. The parameter r is decreased until it reached the left end of the segment. A derailment occurs and the system reaches segment S_2.

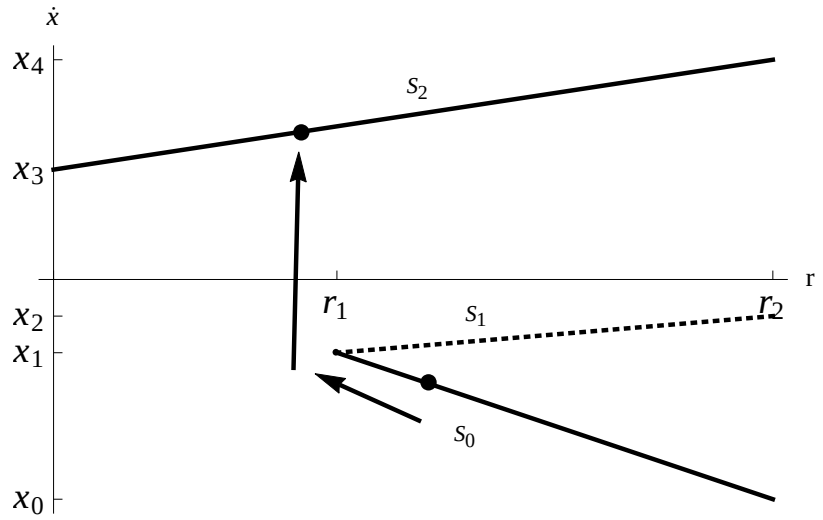


Figure 4.2: Decreasing the parameter r in the bifurcation diagram. As we can observe the system goes from Segment S_0 to S_2. The system goes through to the intersection point of segments S_0 and S_1 and continues to segment S_2

4.3. Qualitative Control Planning

The simulation algorithm derives the qualitative behavior of a system when a given sequence of actions is applied to a given initial state. The *planning problem* for a dynamical system is that of determining the inverse mapping: given an initial state and a goal state, find a sequence of actions that will move the system from an initial to a goal state, if it is possible. More specifically, the problem solved by the planning algorithm is finding a descriptor D that takes the system from a given initial state (i.e., a segment and an r value) to a given final state in the QBD.

The planning algorithm (Algorithm 12) receives as input a queue, a final state, and a QBD. The queue holds a representation of the plans created so far. Every object inserted in the queue has the form $\{State_{current}\ States,\ a_{next},\ Path\}$. Where $State_{current}$ ($\{S_{current},\ r_{current}\}$) is the actual position in the system. a_{next} is the next action (dynamic)

to be performed from $State_{current}$ (i.e., perturbation). *States* are the qualitative states previously visited by the current plan, which is used to prevent loops in plans; *Path* is the descriptor of actions taken so far for this plan.

Algorithm 12 `Planning`($Queue, FinalState, QBD$)

```
1: if Queue is Empty then
2:   Return Fail
3: else
4:   next ← DEQUEUE(Queue)
5:   S_current ← ObtainSegment(next)
6:   r_current ← ObtainPosition(next)
7:   States_current ← ObtainStates(next)
8:   a_next ← ObtainAction(next)
9:   Path ← ObtainPath(next)
10:  if S_current == S_end and r_current == r_end then
11:    Return Path
12:  end if
13:  if S_current == S_end and S_current is stable then
14:    Path ← Append(Path, a_next)
15:    Return Path
16:  else
17:    if a_next is Nil then
18:      for All actions a in getActions(S_current, r_current, States_current, QBD)
         do
19:        Queue ← ENQUEUE(Queue, {S_current, r_current, States_current, a, Path_current})
20:      end for
21:    else
22:      {S_new, r_new} ← NextState({S_current, r_current}, a_next, QBD)
23:      States_new = Append(States_current, S_current)
24:      Path ← Append(Path, a_next)
25:      for All actions a in getActions(S_new, States_new, QBD) do
26:        Queue ← ENQUEUE(Queue, {S_new, r_new, States_new, a, Path_new})
27:      end for
28:    end if
29:    Return Planning(Queue, {S_end, r_end}, QBD)
30:  end if
31: end if
```

Prior to invoking the algorithm, an object $\{\{S_{initial}, r_{initial}\}, Nil, Nil, Nil\}$ is enqueued on to *Queue*. The algorithm starts by checking the queue. If it is empty, then the algorithm could not find a plan to get to the final position r_{end} and halts (Line 2). If *Queue* is not empty, the algorithm considers the first object from the *Queue*. The algorithm dis-

Algorithm 13 `getActions`$(S_{current}, r_{current}, States_{current}, QBD)$

1: $Actions \leftarrow \{\}$
2: $T \leftarrow$ `TakeTransitions`(QBD)
3: $QS \leftarrow$ `TakeQS`(QBD)
4: $Row \leftarrow$ `TakeRow`$(S_{current})$
5: **if** $\{S_{dest}, \{r_{from}, r_{to}\} \epsilon\ UpT(S_{current})$ `and` $r_{from} < r_{current} < r_{to}$ **then**
6: $Actions \leftarrow$ `Append`$(Actions, +)$
7: **end if**
8: **if** $\{S_{dest}, \{r_{from}, r_{to}\} \epsilon\ DownT(S_{current})$ `and` $r_{from} < r_{current} < r_{to}$ **then**
9: $Actions \leftarrow$ `Append`$(Actions, -)$
10: **end if**
11: **if** $\{S_{dest}\} \epsilon\ LeftT(S_{current})$ **then**
12: $S \leftarrow$ `TakeNextSegment`$(Row, Left)$
13: $Limit \leftarrow$ `TakeSegmentLimit`$(S_{current}, Left)$
14: $NewLandmark \leftarrow$ `CreateNewLandmark`$(Limit, S_{current}, Right, QS)$
15: $Actions \leftarrow$ `Append`$(Actions, NewLandmark)$
16: **end if**
17: **if** $\{S_{dest}\} \epsilon\ RightT(S_{current})$ **then**
18: $S \leftarrow$ `TakeNextSegment`$(Row, Right)$
19: $Limit \leftarrow$ `TakeSegmentLimit`$(S_{current}, Right)$
20: $NewLandmark \leftarrow$ `CreateNewLandmark`$(Limit, S_{current}, Right, QS)$
21: $Actions \leftarrow$ `Append`$(Actions, NewLandmark)$
22: **end if**
23: `Return` $Actions$

continues its search in the current direction if the current segment $S_{current}$ is contained in *States*, since a loop has been detected. If the system has reached the final position r_{end}, the algorithm will return *Path* of the current search object (Line 11). If the system is in the final qualitative segment, and this segment is stable, the algorithm needs only to move to the final parameter value. The algorithm adds the action to the final position (r_{end}) to *Path* and returns it (Line 15).

If the system is not in the final qualitative segment, the planning algorithm updates its state to S_{next} and r_{next} within the qualitative bifurcation diagram by calling procedure *NextState* when a_{next} is not Nil (Line 17); the variable a_{next} is only Nil on the first call to *Planning* from the initial state. The function *getActions* searches in QBD for neighbor segments that the system could reach from the current segment, referring to the $Transitions$ of the QBD (Line 18). The function searches from the current state to find possible actions that will change the qualitative state of the system. For example, if the system is in a stable segment, applying a positive or negative perturbation will not be useful, since the system will simply return to the segment where the perturbation took place. For every possible direction and action, the algorithm pushes a new object onto the *Queue* that includes the new action a to be taken (Line 19). After this has been done, the algorithm *Planning* is called recursively with the updated *Queue* to start the same process again (Line 29). By placing possible plan states in the queue and considering them in that order, the planning algorithm executes a breadth-first search for a control plan, guaranteeing a shortest plan in terms of the number of control operations.

The function *getActions* (Algorithm 13) defines the possible actions defined in the transitions table that the planning may perform from the current state. If exists a perturbation transition defined in T, append a positive perturbation (Line 6); same case for a negative perturbation (Line 9). Another case is if there is a Left transition is defined in T for the current state (Line 13), the algorithm creates a new landmark just beyond the left end of the current segment; and append the new landmark to Actions (Line 15). The same process is performed to Right Transition (Line 21).

4.3.1. Planning Example

To illustrate the planning algorithm let us use the same qualitative bifurcation diagram used in the simulation algorithm (the QBD is described in Chapter 3, Figure 3.4). The planning algorithm determines the parameter changes in order to arrive from an initial state to a final destination in the QBD. It receives as input the final destination (goal state) and a queue. For this example lets define the goal state in segment S_2 at r_3 and the first object of the queue is defined as $\{S_0, r_3, Nil, Nil, Nil\}$.

When the queue is empty the algorithm finishes without finding a path. Since the queue is not empty, the algorithm takes the first object of the queue and establishes $S_{current} = S_0$ and $r_{current} = r_3$. At the beginning of the algorithm a_{next} is Nil, for that reason the algorithm enqueues all the actions obtained from the function *getActions*. This function checks the row of the segment in the transition table, where all the actions that the segment could arrive to are defined.

While checking the row for segment S_0. Planning finds that the only possible action is to move the system to the left. If we decrease (move the parameter to the left) the parameter r to a value less than r_1, a *Falloff* will occur and the system will reach segment S_2. For decreasing the parameter r, and move the system to segment S_2, a new landmark needs to be created (less than r_1). A new landmark is created (r_{10}) and inserted in QS, between landmarks r_4 and r_1. A new object is inserted in the queue; this new object has the form $\{\{S_0, r_3\}, r_{10}, Nil, Nil\}$ and the algorithm calls itself recursively.

The algorithm starts taking the first object of the queue and establishes $S_{current} = S_0$, $r_{current} = r_3$, and $a_{next} = r_10$. The algorithm executes the action with the function $PerformAction$ (move to r_{10}) and updates $S_{new} = S_2$, $r_{new} = r_{10}$, $States = \{S_0\}$, and $Path = \{r_{10}\}$.

With this action the algorithm has arrived to the goal segment (S_2). At this point, the function *getAction* only returns the goal landmark, and establishes it as the new action. The algorithm enqueues $\{S_2, r_{10}, \{S_0\}, r_3, \{r_{10}\}\}$ and calls itself once more, only to perform the action. With the action performed the system has arrived to the goal state. The algorithm finishes by returning the current $Path$ joined with the last action ($\{r_{10}, r_3\}$).

Table 4.3: The output after applying the planning algorithm example

$D = \{r_{10}, r_3\}$
$QS_{final} = \{\{r_0, r_4, r_{10}, r_1, r_1, r_3, r_2\}, \{x_0, x_1, x_2, x_3, x_4\}\}$

The output of the algorithm is the Descriptor and the new QS. Since new landmarks are created during the planning process the Quantitative Spaces need to be updated. Table 4.3 shows the output for this example after applying the planning algorithm.

We tested the output of the planning algorithm in the simulation algorithm to verify graphically that the plan produces the expected results (see Figure 4.3). As we can observe, the system starts from S_0 at r_3 and after applying the actions described in the descriptor $D = \{r_{10}, r_3\}$ the system arrives to the goal state.

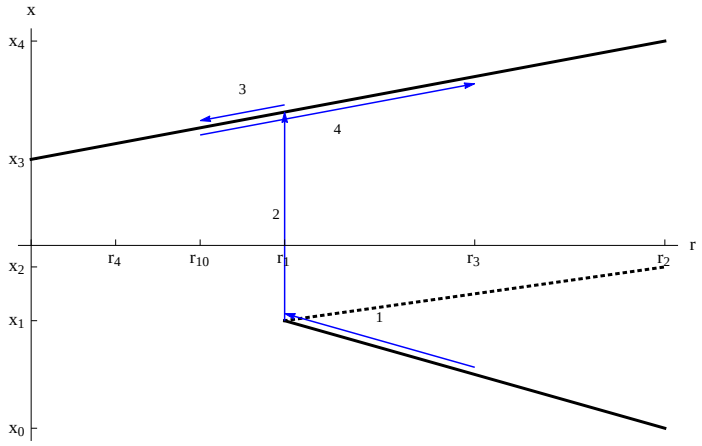


Figure 4.3: Testing the output of the planning algorithm in the simulation algorithm.

4.4. Chapter Conclusions

In this chapter two algorithms were presented. The first algorithm (simulation) describes the behavior of the system when one of its parameters is varied. The second one (planning) finds the actions that the system needs to arrive from one initial to a goal state. In both cases the input (a bifurcation diagram) is a qualitative bifurcation diagram, using the syntax presented in Chapter 3.

Both algorithms are useful to describe the behavior of a dynamical system. The

algorithms use the concepts of flows to predict the evolution of the system. The output of the algorithms is in qualitative terms. At any point of the algorithms numerical values are used.

The simulation and planning algorithms were implemented in Mathematica. The operational explanation of the application is described in Chapter A. Also, some illustrative examples to test the algorithms are presented in Chapter 5

Chapter 5

Results

The three main algorithms proposed in this work, were tested with several examples; previous chapters have shown some basic examples to illustrate the proposed representation and parts of the algorithms. In this chapter we test the algorithms with four examples. The first example corresponds to the system defined by the differential equation $\dot{x} = rx + x^3 - x^5$, the second one corresponds to the Morris-Lecar model, which is a biological neuron model, the third one to the system defined by the differential equation $\dot{x} = 16 + x - x^3$, and the 4th one represents a system defined by the differential equation $\dot{x} = rx + x^3$

5.1. Example 1: A Simple Dynamical System

The first example corresponds to the model defined by the differential equation $\dot{x} = rx + x^3 - x^5$. Figure 5.1 shows the bifurcation diagram corresponding to this dynamical system. For this the example, the input data (BD points) were obtained from the application presented in [Flores10].

5.1.1. Qualitativization

In order to predict the system's behavior, it is necessary to generate the qualitative representation of the BD. Following Algorithm 1 we obtained the qualitative representation of the BD. The graphic version of the qualitative representation is illustrated on Figure 5.2.

From Figure 5.2 we observe that each segment is assigned a symbolic name. The dashed segments are unstable, and the continuous ones are stable.

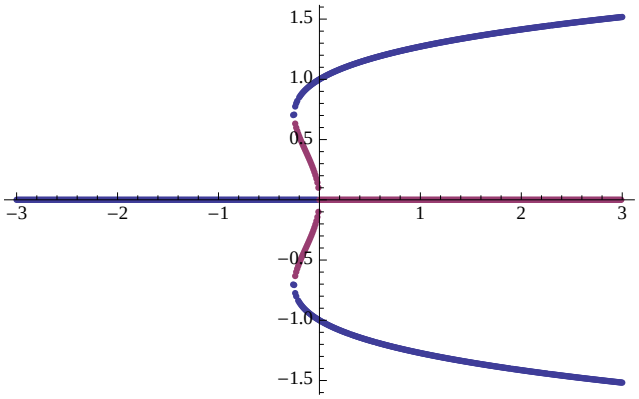


Figure 5.1: Quantitative data of the BD defined by $\dot{x} = rx + x^3 - x^5$.

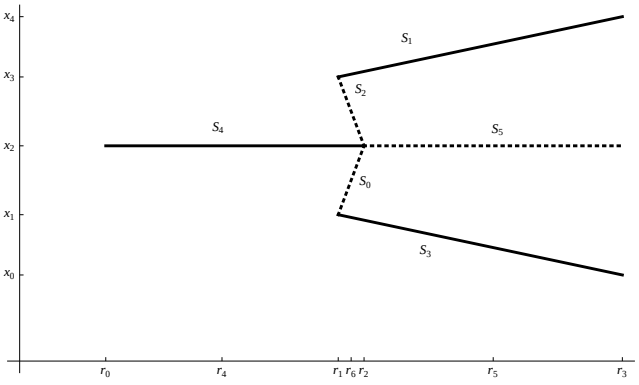


Figure 5.2: Qualitative representation of the BD defined by $\dot{x} = rx + x^3 - x^5$.

Tables 5.1 and 5.2 show the qualitative representation of the BD of Figure 5.2. Table 5.1 describes the variables, the quantity space, and the segments of the BD. Table 5.2 describes all the possible transitions reachable from each segment.

5.1.2. Simulation

Once the QBD was produced by the qualitativization algorithm, we can predict the behavior of the system. To make a prediction, it is necessary to define an initial state and the control sequence (descriptor) applied to parameter r. For this example, the simulation

Table 5.1: Qualitative Representation of the Bifurcation Diagram of Figure 5.2

Variables		$\{r, x\}$
Quantity Space	QS_r	$\{r_0, r_1, r_2, r_3\}$
	QS_x	$\{x_0, x_1, x_2, x_3, x_4\}$
BDS	S_0	$\{\{r_1, x_1\}, \{r_2, x_2\}, \{+, u\}\}$
	S_1	$\{\{r_1, x_3\}, \{r_3, x_4\}, \{+, s\}\}$
	S_2	$\{\{r_1, x_3\}, \{r_2, x_2\}, \{-, u\}\}$
	S_3	$\{\{r_1, x_1\}, \{r_3, x_0\}, \{-, s\}\}$
	S_4	$\{\{r_0, x_2\}, \{r_2, x_2\}, \{0, s\}\}$
	S_5	$\{\{r_2, x_2\}, \{r_3, x_2\}, \{0, u\}\}$
Transitions		See Table 5.2

Table 5.2: Transitions of the Bifurcation Diagram of Figure 5.2

Segment	Up	Down	Left	Right
S_0	$\{S_4,\{r_1, r_2\}\}$	$\{S_3,\{r_1, r_2\}\}$	$\{\}$	$\{\}$
S_1	$\{S_1,\{r_1, r_3\}\}$	$\{S_1,\{r_1, r_3\}\}$	S_4	$\{\}$
S_2	$\{S_1,\{r_1, r_2\}\}$	$\{S_4,\{r_1, r_2\}\}$	$\{\}$	$\{\}$
S_3	$\{S_3,\{r_1, r_3\}\}$	$\{S_3,\{r_1, r_3\}\}$	S_4	$\{\}$
S_4	$\{S_4,\{r_0, r_2\}\}$	$\{S_4,\{r_0, r_2\}\}$	$\{\}$	S_5
S_5	$\{S_1,\{r_2, r_3\}\}$	$\{S_4,\{r_2, r_3\}\}$	S_5	$\{\}$

starts at r_6, located in Segment S_2; the system undergoes the changes described as $D = \{r_4, r_5, -, r_2\}$, which means that parameter r will decrease from r_6 to r_4, then increase from r_4 to r_5, in r_5 there will occur a negative perturbation, and finally the parameter r will decrease to r_2.

Figure 5.3 shows the behavior of the system after applying the changes described by D. As we can see, the system starts at unstable segment S_2. When parameter r moves to r_4, the system first experiences a derailment to segment S_4. When we arrive to segment S_4, the system stays in that segment until it reaches r_4. When r increases to r_5, the system follows the same segment until reaching r_2. In that position segment S_4 ends and segment S_5 begins. The system follows segment S_5 up to r_5, where a negative perturbation occurs. Segment S_5 is unstable, so the system moves to segment S_3 at position r_5. When parameter r decreases, the system follows the same segment until r_2. Table 5.3 presents the output history for the simulation.

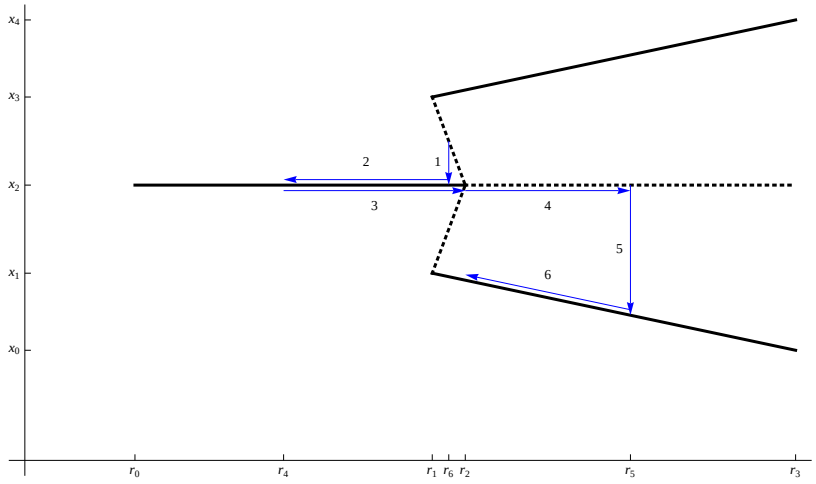


Figure 5.3: Graphic output of the qualitative simulation for Example 1

Table 5.3: The output after applying the simulation algorithm to example 1

	Action	Description
1	$Derailment[S_2, S_4, r_6, -]$	A derailment occurs from S_2 to S_4 at position r_6. The value of x is decremented.
2	$Follow[S_4, r_6, r_4, -, 0]$	The system follows from r_6 to r_4 in the segment S_4, r is decremented (-).
3	$Follow[S_4, r_4, r_2, +, 0]$	The system follows from r_4 to r_2 in segment S_4, r is incremented (+).
4	$Follow[S_5, r_2, r_5, +, 0]$	The system follows from r_2 to r_5 in the segment S_5. r is incremented (+).
5	$Perturbation[S_5, S_3, r_5, -]$	A perturbation occurs from S_4 to S_2 at position r_5. The value of x is decremented (-)
6	$Follow[S_3, r_5, r_2, -, +]$	The system follows from r_5 to r_2 in the segment S_3. r is decremented (-), and x is incremented.

5.1.3. Planning

To illustrate the planning algorithm and derive the descriptor needed to take this system from segment S_1 at $r = r_5$, to a final state in segment S_3 at position r_5. Table 5.4 shows these initial conditions. After applying the planning algorithm, we obtain the descriptor and a new Quantitative Space. During the search process new landmarks are generated; as the algorithm moves between qualitative state segments it generates arbitrary landmarks within those segments. Table 5.6 shows the results of the planning algorithm.

Table 5.4: Input of planning algorithm for example 1

Initial State	$\{S_1, r_5\}$
Final State	$\{S_3, r_5\}$

Table 5.5: The output after applying the planning algorithm in the example 1

$D = \{r_{10}, r_{11}, -, r_5\}$
$QS_{final} = \{\{r_0, r_4, r_{10}, r_1, r_2, r_{11}, r_5, r_3\}, \{x_0, x_1, x_2, x_3, x_4\}\}$

To verify the results of the planning algorithm we submit those results to the simulation algorithm. Table 5.6 shows the results of the simulation algorithm. We can observe from Table 5.4 and Table 5.6 that the QS is not the same. This is because the planning algorithm generates new landmarks at search time. The graphic results can be observed on Figure 5.4.

Table 5.6: Output of the simulation algorithm for the planning example

	Action
1	$Follow[S_1, r_5, r_1, -, -]$
2	$Falloff[S_1, S_4, r_1, -]$
3	$Follow[S_4, r_1, r_{10}, -, 0]$
4	$Follow[S_4, r_{10}, r_2, +, 0]$
5	$Follow[S_5, r_2, r_{11}, -, 0]$
6	$Perturbation[S_5, S_3, r_{11}, -]$
7	$Follow[S_3, r_{11}, r_5, +, -]$

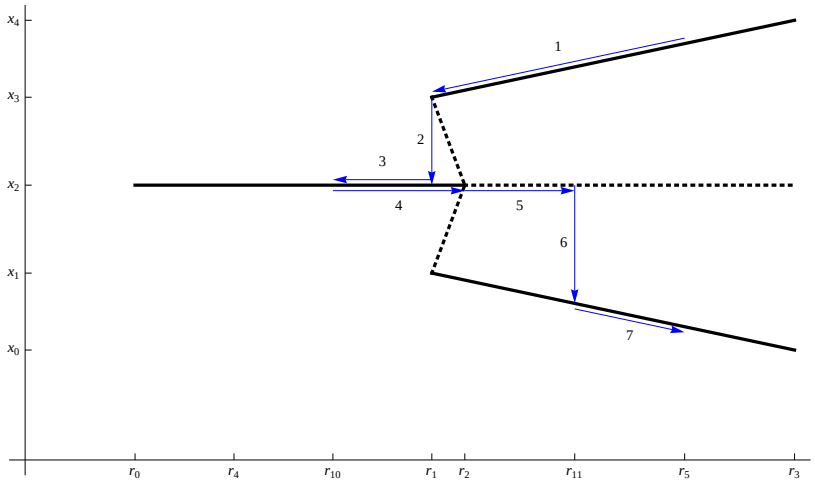


Figure 5.4: Graphic representation of the output of the simulation for the planning example

5.2. Example 2: The Morris-Lecar System

The second example corresponds to the Morris-Lecar Model. This model is a two-dimensional reduced excitation model applicable to systems having two non-inactivating voltage-sensitive conductances developed by Catherine Morris and Harold Lecar [Morris81]. This system of equations describes the relationship between membrane potential and the activation of ion channels within a neuron's membrane. The potential depends on the activity of the ion channels, and the activity of the ion channels depends on the voltage [Tsumoto06]. The Morris-Lecar equations are defined by:

$$C\frac{dV}{dt} = I + g_l(E_l - V) + g_k w(E_k - V) + g_{c_a} m_\infty(V)(E_{c_a} - V) \tag{5.1}$$

$$\frac{dw}{dt} = (w_\infty(V) - w)\lambda_w(V) \tag{5.2}$$

5.2.1. Qualitativization

For this example the input data were obtained from the application XPPAUT [Ermentrout03]. Figure 5.5 shows the bifurcation diagram corresponding to this dynamical system. As mentioned before, to predict the system's behavior, it is necessary to generate the qualitative representation of the BD. Following Algorithm 1 we obtain the qualitative representation of the BD. The graphic version of the qualitative representation is illustrated on Figure 5.6. The dashed segments are unstable, and the continuous ones are stable.

Table 5.7 and 5.8 show the qualitative representation of the BD of Figure 5.5. Table 5.7 describes the variables, the quantity space, and the segments of the BD. Table 5.8 describes all the possible transitions that each segment can reach.

5.2.2. Simulation

To perform a simulation it is necessary to define an initial state of the system and the control sequence that the parameter I will go through (descriptor). In this example we start at segment S_4, located at I_{10}. The actions that the system will suffer are defined as $D = \{I_{11}, I_5, -, I_9\}$. I.e., the parameter I will be decreased to I_{11}, then it will be increased to I_5. A negative perturbation occurs at that point, to finally increase the parameter to I_9.

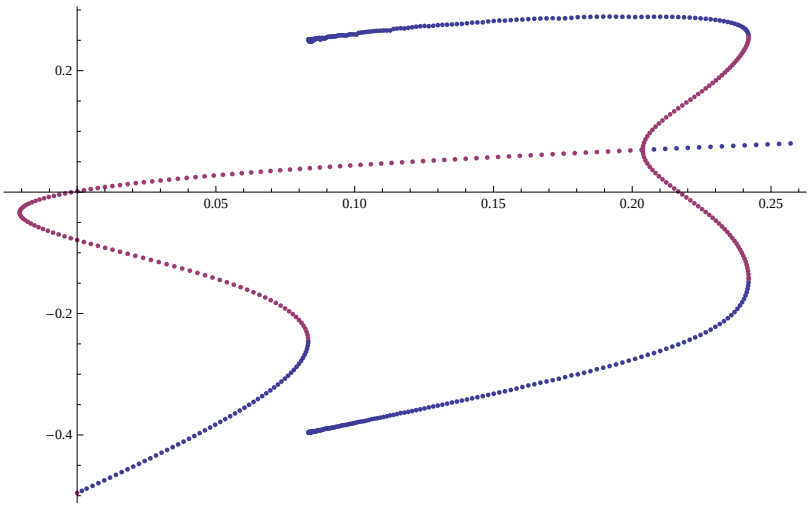


Figure 5.5: Bifurcation diagram of the Morris-Lecar system

Table 5.7: Qualitative Representation of the Bifurcation Diagram of Figure 5.5.

Variables		$\{I, V\}$
Quantity Space	QS_I	$\{I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8\}$
	QS_V	$\{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}, V_{11}\}$
BDS	S_1	$\{\{I_1, V_0\}, \{I_1, V_0\}, \{+, u\}\}$
	S_2	$\{\{I_1, V_0\}, \{I_2, V_2\}, \{+, s\}\}$
	S_3	$\{\{I_2, V_1\}, \{I_7, V_3\}, \{+, s\}\}$
	S_4	$\{\{I_0, V_4\}, \{I_2, V_2\}, \{-, u\}\}$
	S_5	$\{\{I_3, V_5\}, \{I_4, V_6\}, \{+, u\}\}$
	S_6	$\{\{I_4, V_6\}, \{I_6, V_7\}, \{-, u\}\}$
	S_7	$\{\{I_6, V_7\}, \{I_6, V_7\}, \{-, s\}\}$
	S_8	$\{\{I_6, V_7\}, \{I_7, V_3\}, \{-, u\}\}$
	S_9	$\{\{I_0, V_4\}, \{I_3, V_5\}, \{+, u\}\}$
	S_{10}	$\{\{I_6, V_7\}, \{I_8, V_8\}, \{+, s\}\}$
	S_{11}	$\{\{I_6, V_7\}, \{I_7, V_{10}\}, \{+, u\}\}$
	S_{12}	$\{\{I_2, V_9\}, \{I_2, V_9\}, \{-, s\}\}$
	S_{13}	$\{\{I_2, V_9\}, \{I_5, V_{11}\}, \{+, s\}\}$
	S_{14}	$\{\{I_5, V_{11}\}, \{I_7, V_{10}\}, \{-, s\}\}$
Transitions		See Table 5.8

Figure 5.7 shows the behavior of the system after applying the changes described by D. Table 5.9 presents the output history for the simulation.

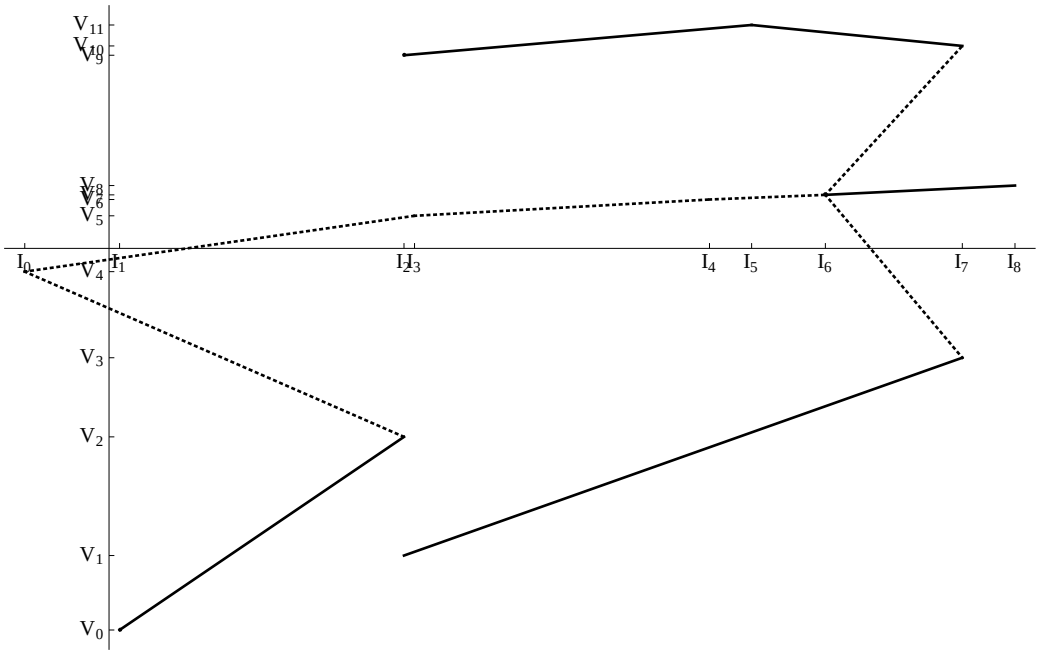


Figure 5.6: Qualitative representation of the bifurcation diagram of the Morris-Lecar system

Table 5.8: Transitions of the Bifurcation Diagram of Figure 5.6.

Segment	Up	Down	Left	Right
S_1	{}	{}	{}	{}
S_2	$\{S_2,\{I_1,I_2\}\}$	$\{S_2,\{I_1,I_2\}\}$	{}	S_3
S_3	$\{S_3,\{I_2,I_7\}\}$	$\{S_3,\{I_2,I_7\}\}$	S_2	S_{10}
S_4	{}	$\{S_2,\{I_1,I_2\}\}$	{}	{}
S_5	$\{S_{13},\{I_3,I_5\}\}$	$\{S_3,\{I_3,I_5\}\}$	{}	{}
S_6	$\{S_{14},\{I_5,I_6\}\}$	$\{S_3,\{I_5,I_6\}\}$	{}	{}
S_7	$\{S_7,\{I_6,I_6\}\}$	$\{S_7,\{I_6,I_6\}\}$	{}	S_{10}
S_8	$\{S_{10},\{I_6,I_7\}\}$	$\{S_3,\{I_6,I_7\}\}$	{}	{}
S_9	{}	{}	{}	{}
S_{10}	$\{S_{10},\{I_6,I_8\}\}$	$\{S_{10},\{I_6,I_8\}\}$	S_7	{}
S_{11}	$\{S_{14},\{I_6,I_6\}\}$	$\{S_{10},\{I_6,I_7\}\}$	{}	{}
S_{12}	$\{S_{12},\{I_2,I_2\}\}$	$\{S_{12},\{I_2,I_2\}\}$	{}	S_{13}
S_{13}	$\{S_{13},\{I_2,I_5\}\}$	$\{S_{13},\{I_2,I_5\}\}$	S_{12}	S_{14}
S_{14}	$\{S_{14},\{I_3,I_7\}\}$	$\{S_{14},\{I_3,I_7\}\}$	S_{13}	S_{10}

5.2.3. Planning

For the planning example it is necessary to specify initial and final states. For this example, we start at segment S_{13} at I_3, and our final destination is segment S_{10} at position I_9

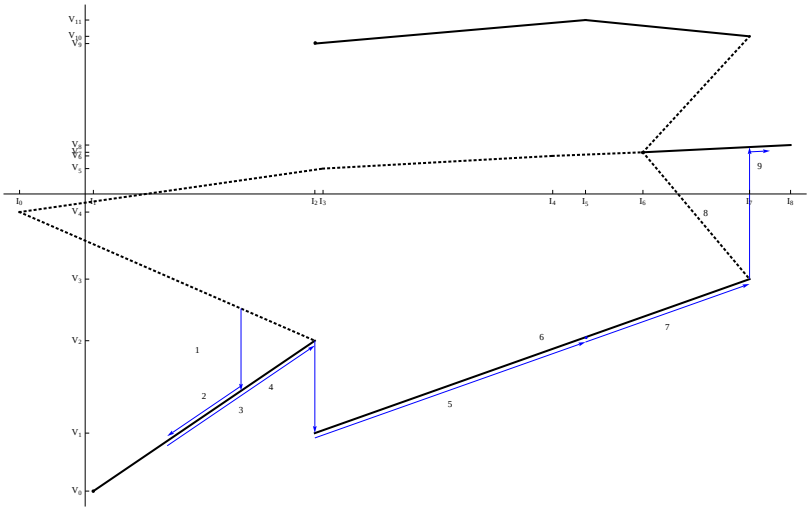


Figure 5.7: Graphic output of the qualitative simulation for Morris-Lecar Model Example

Table 5.9: The output after applying the simulation algorithm to example 1

	Action	Description
1	$Derailment[S_4, S_2, I_{10}, -]$	A derailment occurs from S_4 to S_2 at position I_{10}. The value of v is decremented.
2	$Follow[S_2, I_{10}, I_{11}, -, -]$	The system follows from I_{10} to I_{11} in the segment S_2, I is decremented (-).
3	$Follow[S_2, I_{11}, I_2, +, +]$	The system follows from I_{11} to I_2 in segment S_2, I is incremented (+).
4	$FallOff[S_2, S_3, I_2, -]$	The system falloffs from S_2 to S_3 at the position I_2.
5	$Follow[S_3, I_2, I_5, +, +]$	The system follows from I_2 to I_5 in segment S_3, I is incremented (+).
6	$Perturbation[S_3, S_3, I_5, -]$	A perturbation occurs from S_3 to S_3 at position I_5.
7	$Follow[S_3, I_5, I_7, +, +]$	The system follows from I_5 to I_7 in the segment S_3. I is incremented (+), and V is incremented.
8	$FallOff[S_3, S_{10}, I_7, +]$	The system falloffs from S_3 to S_10 at the position I_7.
9	$Follow[S_{10}, I_7, I_9, +, +]$	The system follows from I_7 to I_9 in the segment S_10. I is incremented (+), and V is incremented (+).

(Table 5.10 shows these initial conditions). After applying the planning algorithm, we obtain the descriptor and a new Quantitative Space. During the search process new landmarks are

generated; as the algorithm moves between qualitative state segments, it generates arbitrary landmarks within those segments. Table 5.10 shows the results of the planning algorithm.

Table 5.10: Input of the planning algorithm in the Morris-Lecar Model example

Initial State	$\{S_{13}, I_3\}$
Final State	$\{S_{10}, I_9\}$

Table 5.11: Output of the planning algorithm in the Morris-Lecar Model example

$D = \{I_{20}, I_{21}, I_9\}$
$QS_{final} = \{\{I_0, I_1, I_{,11}, I_{10}, I_2, I_3, I_4, I_5, I_{20}, I_6, I_7, I_{21}, I_9, I_8\},$
$\{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}, V_{11}\}\}$

To verify the results of the planning algorithm we submitted them to the simulation algorithm. Table 5.12 shows the results of the simulation algorithm. The graphic results can be observed on Figure 5.8.

Table 5.12: Output of the simulation algorithm for the Morris-Lecar in the planning example

	Action
1	$Follow[S_{13}, I_3, I_5, +, +]$
2	$Follow[S_{14}, I_5, I_{20}, +, -]$
3	$Follow[S_{14}, I_{20}, I_7, +, -]$
4	$Falloff[S_{14}, S_{10}, r_7, -]$
5	$Follow[S_{10}, I_7, I_{21}, +, +]$
6	$Follow[S_{10}, I_{21}, I_9, +, +]$

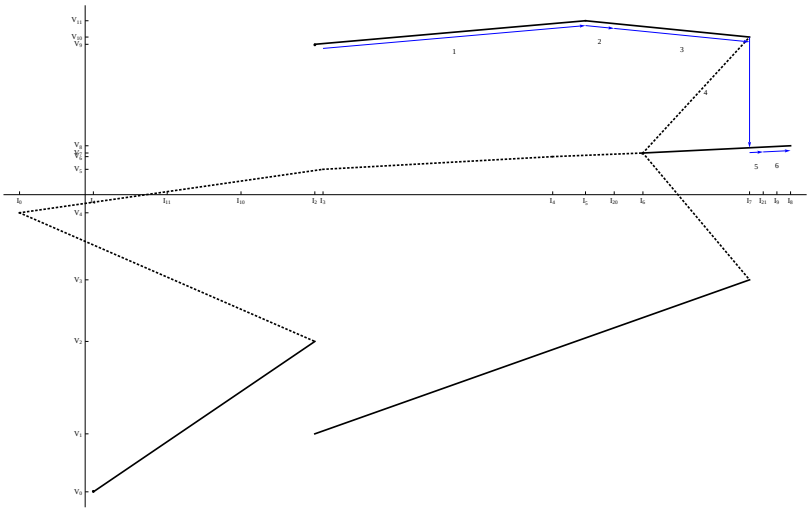


Figure 5.8: Graphic representation of the execution of the results of the planning algorithm for the Morris-Lecar Model

5.3. Example 3

The third example corresponds to the model defined by the differential equation $\dot{x} = 16 + x - x^3$. Figure 5.9 shows the bifurcation diagram corresponding to this dynamical system. For this the example, the input data (BD points) were obtained from the application presented in [Flores10]. After translating the quantitative data into the qualitative representation performed by the QRG algorithm. We observe that the qualitative representation of the dynamical system is formed by three MS, two stables and one unstable. The qualitative diagram is illustrated in Figure 5.10. The qualitative representation is shown in Table 5.13.

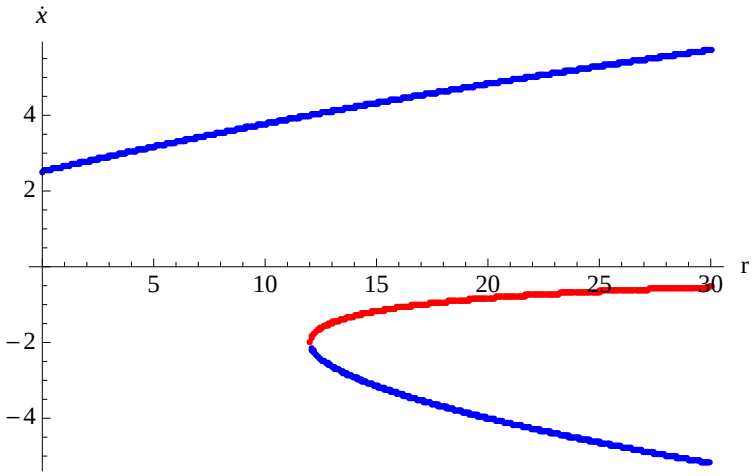


Figure 5.9: Bifurcation diagram defined by $\dot{x} = 16 + rx - x^3$. Blue dots correspond to stable fixed points and red dots to unstable ones.

Table 5.13: Qualitative Representation of the Bifurcation Diagram for Example 3.

Variables		$\{r, x\}$
Quantity Space	QS_r	$\{r_0, r_1, r_2\}$
	QS_x	$\{x_0, x_1, x_2, x_3, x_4\}$
BDS	S_0	$\{\{r_1, x_1\}, \{r_2, x_0\}, \{-, s\}\}$
	S_1	$\{\{r_1, x_1\}, \{r_2, x_2\}, \{+, u\}\}$
	S_2	$\{\{r_0, x_3\}, \{r_2, x_4\}, \{+, s\}\}$
Transitions		See Table 5.14

Let us show some simulations and a control planing for this third example, Table

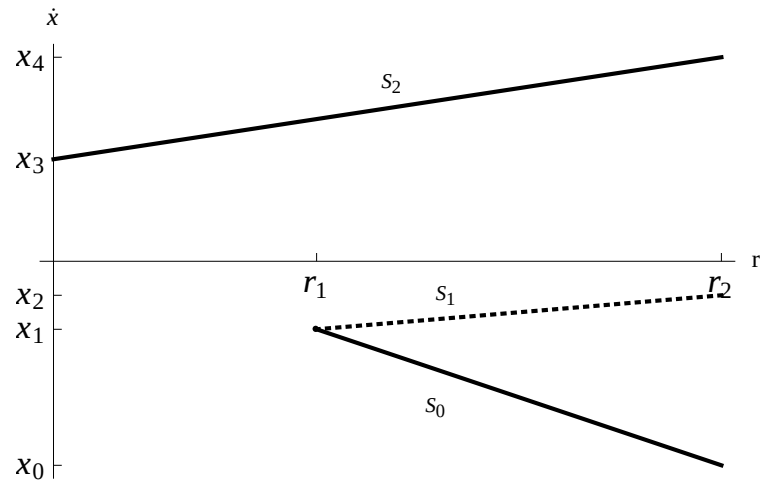


Figure 5.10: Bifurcation diagram of the third example divided in monotonic segments. The black lines correspond to a stable segment and the discontinues line correspond to an unstable segment.

Table 5.14: Transitions of the Bifurcation Diagram for Example 3.

Segment	Up	Down	Left	Right
S_0	$\{S_0, \{r_1, r_2\}\}$	$\{S_0, \{r_1, r_2\}\}$	S_2	$\{\}$
S_1	$\{S_2, \{r_1, r_2\}\}$	$\{S_0, \{r_1, r_1\}\}$	$\{\}$	$\{\}$
S_2	$\{S_2, \{r_0, r_2\}\}$	$\{S_2, \{r_0, r_2\}\}$	$\{\}$	$\{\}$

5.15 shows the inputs for the different simulations. Notice that for all examples (simulations and planning) the Quantitative Space (QS) used is the same. For the first simulation, the simulation algorithm starts from the initial state defined as $Start = \{S_2, r_5\}$, and the variations in the parameter are defined by $D = \{r_3, r_4, r_3\}$. The first action is to move the system from r_5 to r_3, since the actual segment (S_2) is unstable, a *Derailment* occurs; making the system arrive to the segment S_1. After arriving to segment S_1, due the actual segment is stable the system *Follow* until r_3. The next action is going from r_3 to r_4, this mean a decrement in the parameter. The system *Follow* segment S_1 until arrive the left end of the segment (r_1). But, the system has not yet arrive to r_4, therefore the system takes another segment to arrive r_4. The parameter is decremented, then a *Falloff* occurs. The system goes to segment S_3, but has not arrive to r_4. The system *Follow* the actual segment until r_4. Finally the last action is moving the segment from r_4 to r_3. Since the segment S_3 is stable and r_3 is inside the boundary of the segment, the system only *Follow* until r_3. Similar

cases occurs in Simulation 2 and in Simulation 3. The graphically output of the simulation algorithm for the first simulation is illustrated in Figure 5.11. Figure 5.12 and Figure 5.13 shows the graphically output for the simulation 2 and 3 respectively.

Table 5.15: Inputs for the simulation and planning algorithms for example 3

QS	$\{\{r_0, r_4, r_1, r_5, r_3, r_2\}, \{x_0, x_1, x_2, x_3, x_4\}\}$
Simulation 1	
Initial State	$\{S_2, r_5\}$
Descriptor	$D = \{r_3, r_4, r_3\}$
Simulation 2	
Initial State	$\{S_2, r_3\}$
Descriptor	$D = \{r_4\}$
Simulation 3	
Initial State	$\{S_1, r_3\}$
Descriptor	$D = \{r_4, r_3\}$
Planning 1	
Initial State	$\{S_1, r_3\}$
Final State	$\{S_3, r_3\}$

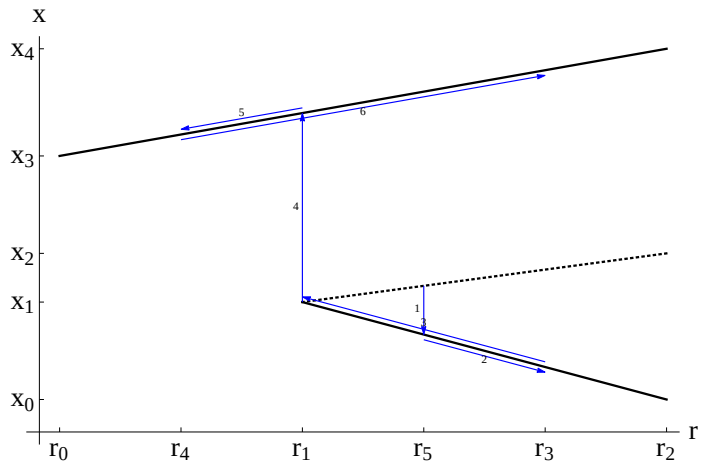


Figure 5.11: Simulation process for the first simulation. The simulation is performed in bifurcation diagram defined in the third example of this thesis. The simulation starts from $\{S_2, r_5\}$, and the changes in the parameter are defined as $D = \{r_3, r_4, r_3\}$. Blue lines shows the path founded after applying the changes defined in D.

Now, let us describe the control planning simulation for this example. The system starts from the initial state $\{S_1, r_3\}$, and the final state is $\{S_3, r_3\}$. Figure 5.14 shows graphically the path (blue lines) found during the planning algorithm. During the planning

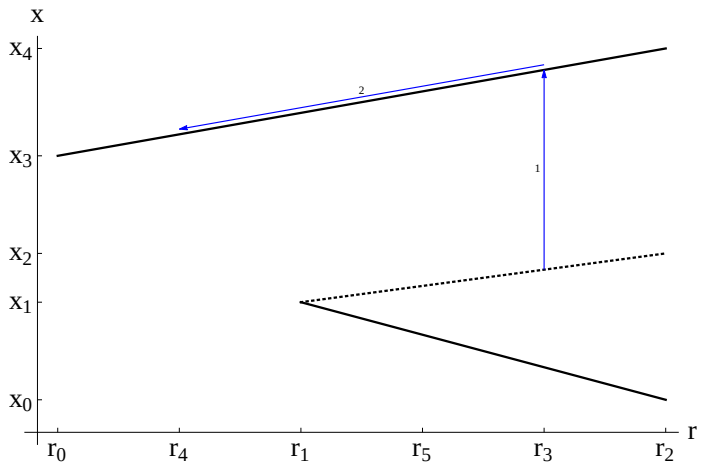


Figure 5.12: Simulation process for the second simulation. The simulation is performed in bifurcation diagram defined in the third example of this thesis. The simulation starts from $\{S_1, r_3\}$, and the changes in the parameter are defined as $D = \{r_4\}$. Blue lines shows the path founded after applying the changes defined in D.

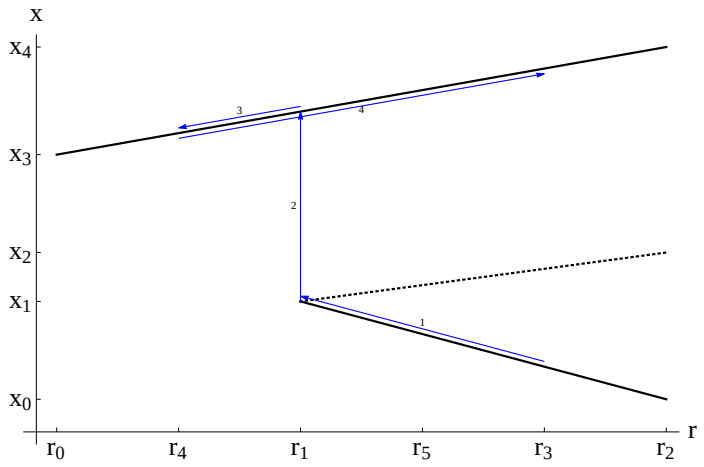


Figure 5.13: Simulation process for the third simulation. The simulation is performed in bifurcation diagram defined in the third example of this thesis. The simulation starts from $\{S_2, r_5\}$, and the changes in the parameter are defined as $D = \{r_4, r_3\}$. Blue lines shows the path founded after applying the changes defined in D.

process new landamarks are created. For this example the landmark r_{10} was created and inserted between r_4 and r_1. The changes in the parameter resulting in the planning process and the new QS are shown in Table 5.16

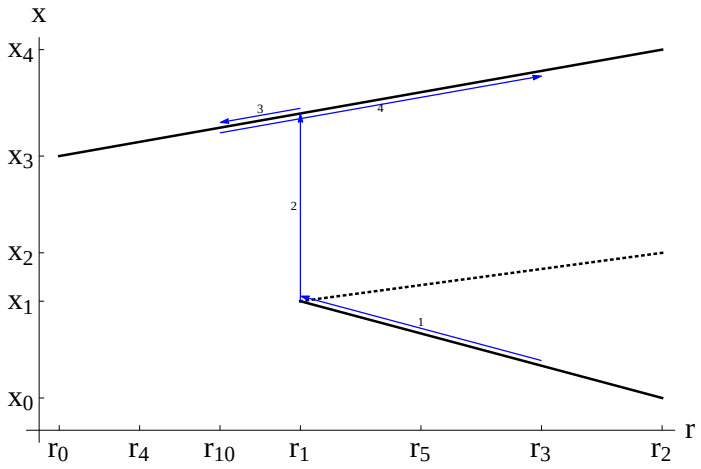


Figure 5.14: Simulation process after applying the output of the control planning process. The simulation is performed in bifurcation diagram defined in the third example of this thesis. The simulation starts from $\{S_1, r_3\}$, and the changes in the parameter are defined by the output of the control planning process. The output founded is $D = \{r_{10}, r_3\}$. Blue lines shows the path founded after applying the changes defined in D.

Table 5.16: The output after applying the planning algorithm in the example 3

$D = \{r_{10}, r_3\}$
$QS_{final} = \{\{r_0, r_4, r_{10}, r_1, r_5, r_3, r_2\}, \{x_0, x_1, x_2, x_3, x_4\}\}$

5.4. Example 4

The 4th example corresponds to the model defined by the differential equation $\dot{x} = rx + x^3$. Figure 5.15 shows the bifurcation diagram corresponding to this dynamical system. For this the example, the input data (BD points) were obtained from the application presented in [Flores10].

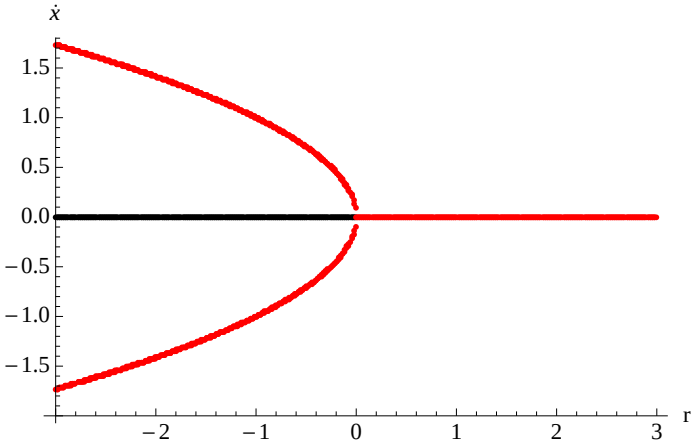


Figure 5.15: Bifurcation diagram defined by $\dot{x} = 16 + rx - x^3$. Blue dots correspond to stable fixed points and red dots to unstable ones.

The qualitative diagram is illustrated in Figure 5.16. The qualitative representation is shown in Table 5.17. As observed in the figure, there are four MS; three of them are unstable and the other one is stable.

Table 5.17: Qualitative Representation of the Bifurcation Diagram for the Example 4.

Variables		$\{r, x\}$
Quantity Space	QS_r	$\{r_0, r_1, r_2\}$
	QS_x	$\{x_0, x_1, x_2\}$
BDS	S_1	$\{\{r_0, x_0\}, \{r_1, x_1\}, \{+, u\}\}$
	S_2	$\{\{r_1, x_1\}, \{r_2, x_1\}, \{0, u\}\}$
	S_3	$\{\{r_0, x_1\}, \{r_1, x_1\}, \{0, s\}\}$
	S_4	$\{\{r_0, x_2\}, \{r_1, x_1\}, \{-, u\}\}$
Transitions		See Table 5.18

Let us show some simulations for this 4th example, Table 5.19 shows the input for simulation and control for Example 4.

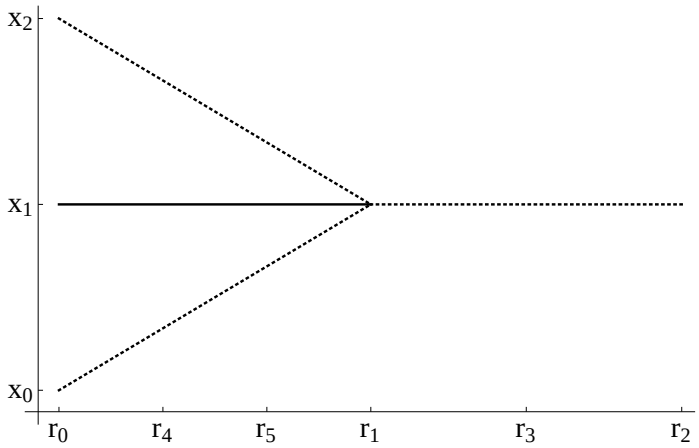


Figure 5.16: Bifurcation diagram of the 4th example divided in monotonic segments. The black lines correspond to a stable segment and the red line correspond to an unstable segment.

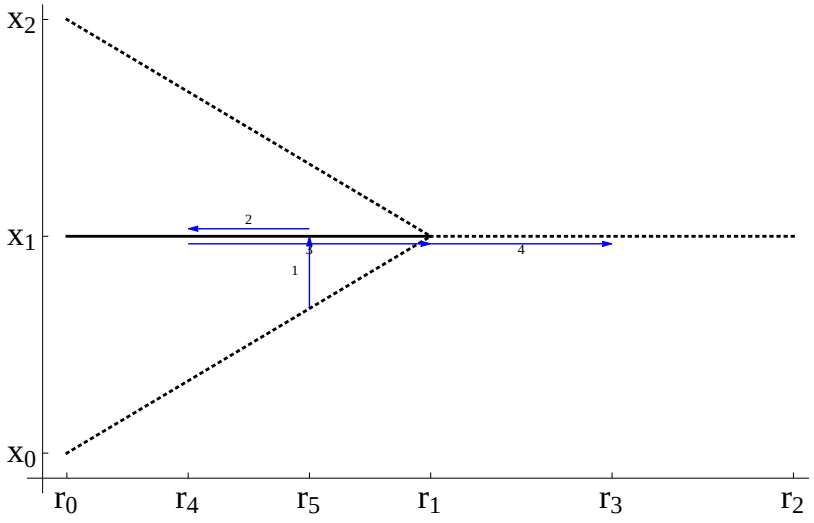


Figure 5.17: Simulation process for the simulation. The simulation is performed in bifurcation diagram defined in the 4th example of this thesis. The simulation starts from $\{S_1, r_5\}$, and the changes in the parameter are defined as $D = \{r_4, r_3\}$. Blue lines shows the path founded after applying the changes defined in D.

Table 5.18: Transitions of the Bifurcation Diagram for the Example 4.

Segment	Up	Down	Left	Right
S_1	$\{S_3, \{r_0, r_1\}\}$	$\{\}$	$\{\}$	$\{\}$
S_2	$\{\}$	$\{\}$	S_3	$\{\}$
S_3	$\{S_3, \{r_0, r_1\}\}$	$\{S_3, \{r_0, r_1\}\}$	$\{\}$	S_2
S_4	$\{\}$	$\{S_3, \{r_0, r_1\}\}$	$\{\}$	$\{\}$

Table 5.19: Inputs for the simulation and planning algorithms for Example 4

QS	$\{\{r_0, r_4, r_5, r_1, r_3, r_2\}, \{x_0, x_1, x_2\}\}$
Simulation	
Initial State	$\{S_1, r_5\}$
Descriptor	$D = \{r_4, r_3\}$
Planning	
Initial State	$\{S_4, r_4\}$
Final State	$\{S_2, r_3\}$

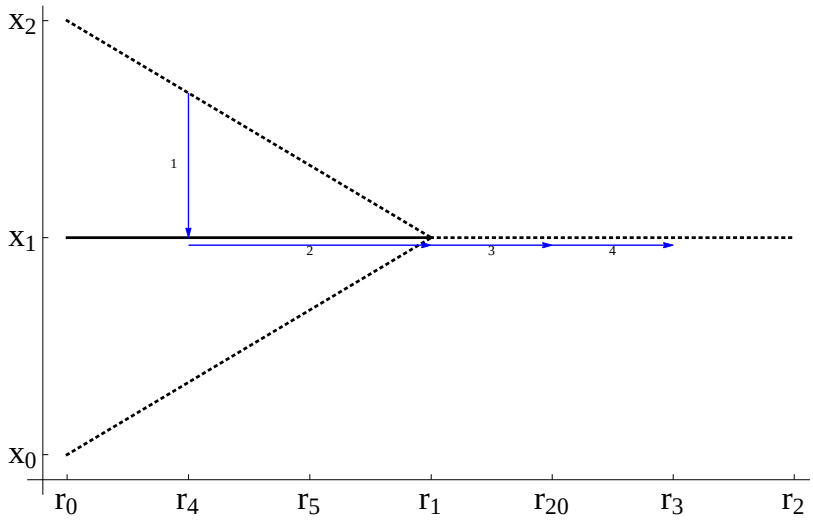


Figure 5.18: Simulation process after applying the output of the control planning process. The simulation is performed in bifurcation diagram defined in the 4th example of this thesis. The simulation starts from $\{S_4, r_4\}$, and the changes in the parameter are defined by the output of the control planning process. The output founded is $D = \{-, r_{20}, r_3\}$. Blue lines shows the path founded after applying the changes defined in D.

5.5. Chapter Conclusions

This chapter presents four examples of dynamical systems and their bifurcation diagrams. The BDs were used to test the algorithms presented in previous chapters. We are able to infer the evolution of the system with the simulation algorithm. We were also able to define the changes to apply to the system in order to move the system to a goal state using the planning algorithm.

Both algorithms use the qualitative representation presented in Chapter 3, produced using the QRG algorithm. The output of the algorithms are expressed in qualitative terms. During the simulation or planning processes no numerical value was used to perform any of these task.

More examples were tested (about ten of them), but I consider that the examples included in this chapter illustrate the main of the three main algorithms. The simulation and planning algorithm have problems when the dynamical system has as solution imaginary roots. In those cases the fixed points do not fulfill the alternation of stabilities theorem described in [Flores06]. Figure 5.19 shows an example of a bifurcation diagram produced using XPPAUT, where two unstable segments are neighbors. In that case, if the state of the system lies between them, to what state the system transition to? that is both fixed points are repellers!. Figure 5.19 also show two segments too close together. In that case, the segmentation algorithm cannot distinguish one segment from another. These are the two main problems we are not able to solve with the proposed methodology.

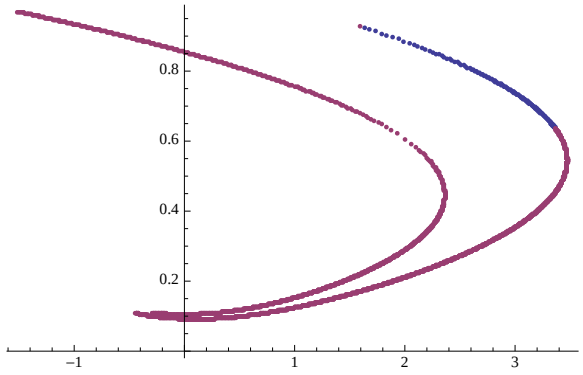


Figure 5.19: A bifurcation Diagram produced by XPPAUTO that shows the two main problems the proposed methodology cannot solve.

Chapter 6

Conclusions

A dynamical system is a rule describing the evolution with time of a point in a given set. This rule can be specied by very dierent means like ordinary dierential equations, iterated maps, partial dierential equations, or cellular automata. Therefore, when we talk about a dynamical system, we might first think of it in terms of differential equations, numerical methods, etc. To describe the systems' behavior we can use a computer to solve the problem numerically. Thinking about solving numerical problems discarding numerical information may sound irrational, but consider that the human brain makes decisions about dynamical systems without any numerical operation.

The idea of qualitative reasoning is to create non-numerical descriptions of physical systems and their behaviors, preserving important behavioral properties and qualitative distinctions. In this thesis, I describe the behavior of a dynamical system discarding all numerical values, instead qualitative values were used. When investigating related work in the area of qualitative reasoning, we found that there are works in qualitative reasoning in the area of dynamical systems but none of them address the bifurcation analysis problem.

6.1. General Conclusions

In this thesis, I present three algorithms that together allow us to describe and control the behavior of a dynamical system using qualitative reasoning. The first algorithm transforms a bifurcation diagram of a given dynamical system, described in numerical terms,

into a qualitative representation of a bifurcation diagram. The second and third algorithms, simulation and planning, use this qualitative representation of the system to define the evolution of a system or define the parameter changes to arrive to a goal state.

The goal of qualitative reasoning is to develop representation and reasoning methods that enable computer programs to reason about the behavior of physical systems, without precise quantitative information. The first algorithm generates the representation that discards the precise quantitative information; the second and third algorithm are the reasoning methods that enable a computer to reason about bifurcation diagrams of physical systems.

The first algorithm (QRG) generates a qualitative representation of a bifurcation diagram (a QBD) from a given quantitative representation. This algorithm starts from a quantitative bifurcation diagram and segments it into qualitative regions or Monotonic Segments (MSs). All the points contained in a region have the same qualitative behavior. A qualitative filter is used to obtain the qualitative slope of each point in the diagram. The qualitative filter is necessary to deal with noise present in the numeric data. With the qualitative slope, it is possible to break the diagram into Monotonic Segments. Due to imprecision in the given quantitative information, some adjustments to the segmentation are made. Once defined the MSs, all the next possible states are define by an interconnected net call *links*.

The qualitative representation proposed in this work was tailor made to keep the relevant information and discard everything else; in this process, new information is generated (*links*). This qualitative representation was developed based on previous works. I took some ideas from different previous works, like the ones presented in [Flores06], [Kuipers94], [Forbus84], [Johan De Kleer84]. Some ideas were taken from these works and adapted for this particular problem in order to capture the essential features of the system.

The links define the next state of the system form any point of the bifurcation diagram, therefore are the relevant part of the simulation and planning algorithms. Without them, the algorithms (simulation and planning) should have to implement new methods to define the next state of the system. Since the next states are non defined, the simulation will have to implement new methods that determine the next state. During simulation and

planning the application works without any numerical information, therefore the implementation of the next state methods will be harder to implement.

In discrete systems, the future state of the system can be analyzed in terms of individual component behavior. Envisionment involves predicting the series of qualitative states that will result from any perturbation to the system[deKleer82]. The transition table defines the next state that will result from any change in the system. This table adds the envisonment to the system.

As already mentioned, both algorithms work without any numerical information. What both of them use is the transitions table (envisionment) and the quantitative space. The quantitative space are two sets of landmarks (for x_s and r_s) that are ordered by magnitude. This order establishes the relation of magnitude and allows us to discard the numerical information for the simulation and planning algorithms.

QRG uses different techniques to segment the information contained in the bifurcation diagram; selecting the data observed from below, or the use of the qualitative filter to define the slope. But once we segment the bifurcation diagram, it is necessary to put everything back together in qualitative terms. Finally, we add the links to the representation to facilitate the implementation of the next two algorithms.

Now let us talk about the simulation and planning algorithms, given a QBD, the simulation algorithm needs only to follow paths described in the transitions table. It is not only possible to simulate the behavior of the system given an initial situation and dynamics descriptor, but also to determine a control sequence (dynamics descriptor) that takes a system from an initial to an end point. Given an initial state and a dynamics descriptor, the simulation describes the system's behavior in qualitative terms, i.e., how it moves from qualitative state (segment) to qualitative state, driven by the dynamics descriptor. Through a typical problem-solving search process, the planning algorithm is able to discover control sequences that can take a system from an initial to a final qualitative state, when such a path exists.

In terms of complexity, QBD traverses D; during this traversal each action can only result in a finite number (at most the number of segments) of steps. The worst case complexity of the algorithm is $O(|D| \times |S|)$, but it would expect be of the order of $O(|D|)$.

(D is the action descriptor list and S is the set of MSs in the BD). The planning algorithm performs a breath first search, so will it find a path if it exists. Planning will visit every segment of the QBD, trying all actions at each segment. By recording the visited states, we can avoid the exponential growth of this search when we repeat searches starting at previously visited segments. So, by pruning the search procedure, the resulting complexity for the planning algorithm is of order $O(|S|)$.

The system QBD has been tested with a dozen of continuous systems, taken from books and from application areas like Electrical Engineering and Biology. In this report we are presenting one example from Strogatz's book and one application example from the area of Biology.

The methodology and all three algorithms have been implemented in Mathematica [Wolfram10]; all the output and plots presented for the examples were produced by the implementation.

The work presented in this thesis proves that sometimes, and for some particular problems, there is not need to perform numerical computations to solve dynamic systems problems. Instead there are alternatives like qualitative reasoning that allow to achieve good conclusions for dynamic systems problems.

We observe that by taking the essential features of the system, using them in an appropriate way, we can arrive to useful conclusions. Currently there is no research work in this direction, or a close approach.

In summary, in this thesis we deal with the problem of bifurcation diagram analysis; dealing with it in qualitative terms. For that reason a qualitative representation (QR) was designed without any previous work on the problem. In the qualitative representation a envisionment (links) was added. Once the QR was designed, the methods that analyze the bifurcation diagrams were created (simulation and planning algorithms). The implementation of the three algorithms help to automatize the bifurcation diagram analysis. This automation facilitate the analysis of a dynamical system.

6.2. Future Work

In this thesis we perform a bifurcation diagram analysis in qualitative terms. For future work we can consider different directions:

- In this work we only consider problems with one state variable, and one parameter. In future work, we intend to generalize the dimensionality for the qualitative representation of a bifurcation diagram with two or more parameters or state variables.

- Every work can be improved if you reconsider the problem one more time. We could improve the qualitative representation developed in this work by deleting the componetes that are not used in the simulations, and adapt the algorithms to work with this improved representation. With the improved representation we could intend to solve problems with a bigger dimensionality.

- In this work during the simulation and planning methods we discarded all numerical information. In real problems, performing a bifurcation diagram analysis we may want to use real values. For example increasing the parameter until a real value instead of a symbolic name (landmark). We may think of adapting the algorithms to work with quantitative and qualitative values. We could perform a semi-qualitative bifurcation analysis.

- In another direction, we can think about solving other kind of problems using qualitative reasoning (not solving numerically). We can think of translating the problem in qualitative terms and from there infer useful conclusions.

Apendix A

QBD A user's manual

QBD is the implementation of the three algorithms proposed in the previous chapters. The first algorithm corresponds to the qualitative representation generation. This algorithm captures the essential features of the bifurcation diagram and translates them to the proposed qualitative representation. The second algorithm describes the behavior of the system when one of its parameters changes. And the third algorithm describe the parameter changes to move the system from an initial to a final state.

Chapter describes how to operate the application. We will describe the structure of the input, and the structure of the output, and how to run the application. The application is implemented in Mathematica 8.0 [Wolfram10].

A.1. Data Input

A dynamical system is usually modeled by a set of differential equations. The solution of the differential equations are the fixed points of the system for one particular value(s) of the parameter(s). Varying one or more of the system's parameters and solving the system we obtain a set of fixed points. This set of fixed points form the bifurcation diagram. As mention in previous chapters, the input of the application QBD starts with the set of fixed points of the bifurcation diagram. To obtain the fixed points for our experiments we used two different sources, the system implemented by Flores [Flores10], and XPPAUT [Ermentrout03]. Routines were implemented to standardize the input to a single

format.

The input format of the QBD application are two arrays of points. Each array corresponds to the stable and unstable fixed points, respectively. Every point of the array is constituted by its x and its y coordinates, for example:

- $StableFixedPoints = \{\{x_1, y_1\}, \{x_2, y_2\}, ..., \{x_n, y_n\}\}$ for n stable fixed points.
- $UnstableFixedPoints = \{\{x_1, y_1\}, \{x_2, y_2\}, ..., \{x_m, y_m\}\}$ for m unstable fixed points.

Figure A.1 illustrates the format for the first 20 elements of the unstable fixed array from one example.

```
UnstableFixedPoins[[1 ;; 20]]
{{12., -2.}, {12.05, -1.87371}, {12.1, -1.82306}, {12.15, -1.78487}, {12.2, -1.75313},
 {12.25, -1.72551}, {12.3, -1.70083}, {12.35, -1.67836}, {12.4, -1.65766}, {12.45, -1.63839},
 {12.5, -1.62033}, {12.55, -1.6033}, {12.6, -1.58715}, {12.65, -1.57179}, {12.7, -1.55712},
 {12.75, -1.54307}, {12.8, -1.52958}, {12.85, -1.5166}, {12.9, -1.50408}, {12.95, -1.49198}}
```

Figure A.1: The format of the first 20 elements of the unstable fixed points array from one experiment.

A.2. Qualitative Representation Generation

To perform the simulations of the system behavior, it is necessary to transform the bifurcation diagram to the qualitative representation defined in Chapters 3 . To produce the qualitative representation it is necessary to load the implemented libraries. The libraries needed to run the complete application are:

- *QualitativeGeneration.m* This library defines the routines to translate the quantitative data of the bifurcation diagram to the qualitative representation defined in previous chapters.
- *SelectData.m* This library defines the necessary routines to break the bifurcation diagram into Monotonic Segments.
- *GenerateLinks.m* This library defines the routines to generate the interconnected network where the continuity of the qualitative segments are defined.

- *FromXppoutToMath.m* This library defines the necessary routines to translate the XPPAUT bifurcation diagram format to the QBD format.
- *Simulacion.m* in this library are defined the necessary routines to perform the simulation of the system.
- *Planning.m* in this library are defined the necessary routines to perform the Control Planning of the system.

Figure A.2 illustrates the libraries loaded to execute the QBD application.

```
In[1]:= rutaDir = NotebookDirectory[];
        << (rutaDir <> "QualitativeGeneration.m");
        << (rutaDir <> "FromXppoutToMath.m");
        << (rutaDir <> "Simulacion.m");
        << (rutaDir <> "SelectData.m");
        << (rutaDir <> "GenerateLinks.m");
        << (rutaDir <> "Planning.m");
        Loaded OK ......
```

Figure A.2: Loaded Libraries of the QBD application.

To generate a qualitative bifurcation diagram it is necessary to run the $GenerateQR[PI, PE]$ function. This function receives two input parameters; the sets of stable and unstable fixed points. The output of this routine is the MSs, the set for links of every segment and the QS. Figure A.3 shows the routine GenerateQR receive as parameters the stable and unstable sets of fixed points, and generates as an exit the MSs, Links of the segments and QS. Figure A.4 illustrate the format of the output.

```
In[28]:= QR = GenerateQR[PI, PE];
         {seg, links, qs} = QR;
```

Figure A.3: Input of qualitative representation generation.

Within the function $GenerateQR[PI, PE]$ are certain parameters that can be varied depending on the problem. The parameters that can be changed are:

- **grid**: This parameter is used to define the distance between every point in the x axis. It is used to discretize the information. For example, if we take the data from

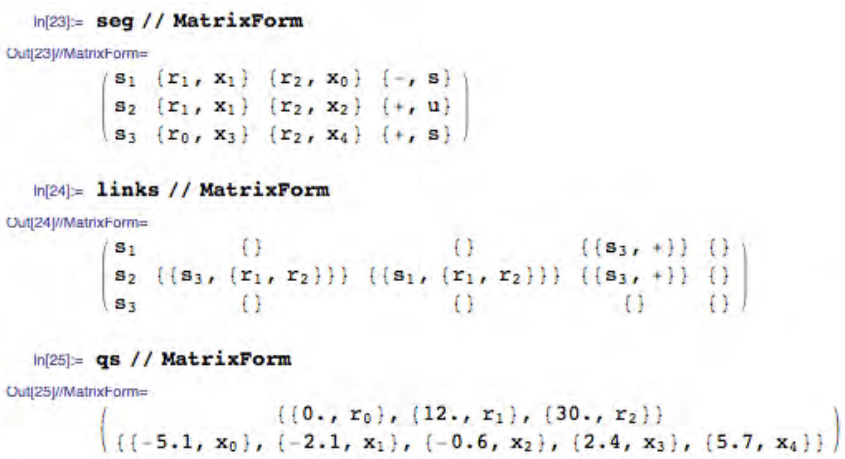


Figure A.4: Output format of qualitative representation generation.

XPPAUT, the data is not uniform. Note that the distance between each point of the x axis depends on the problem.

- **delta**: This parameter is used in Algorithm 1 Line 1 at the time of segmenting the bifurcation diagram. To brake the bifurcation diagram in sets where all the points belong to a function (defined at Section 3.4.2). We take all the points that can be observed from below, but not all of them belong to the segment (as observed in the Figure 3.6). For this cases we need to define a distance from one point to another, to consider if the next point observed belongs to the segment, delta is that distance.

- **hole**: This parameter is also used in Line 1 of Algorithm 1. Since the information is discretized, it may presents noise. Therefore, there may be gaps in the information as observed in Figure 3.7. In this part, the algorithm starts taking points that can be observed from below from the lowest point of the diagram in one direction (left or right). During this process, it may find holes, but it may also start considering points that belong to another segment. (see Figure 3.6). If we take some consecutive points, and these points are discarded because they are too far according to the parameter *delta*, we say there is a hole in the diagram. If the hole is too big, the process is stopped and a new search is started. This parameter is used to define the consecutive number of points to consider before braking the process (define the size of the hole).

- **deltaglue** Algorithm 1, Line 3, the algorithm glues the segments resulting from the previous step. *deltaglue* is the parameter used to define how far the end of a segment is from another segment to be glued. In this case, to glue a segment to another, both segments needs to have the same slope and stability

- **deltaintersec** Used in the same case of the parameter *deltaglue*. But this parameter is used for intersecting segments with different slope or stability.

- **w** This parameter is used to define the window size for the kernel function, at the time of generating the qualitative slope in the qualitative filter.

Figure A.5 shows the parameters inside the routine that can be varied for each problem.

```
grid = 0.0008;          (*Defines the grid for the system*)
delta = 0.5;            (*Used for brake the diagram in MSs*)
hole = 5;              (*Used for brake the diagram in MSs*)
deltaglue = 0.2;        (*Used to join segments*)
deltaintersec = 0.2;    (*Used to join segments*)
w = 8;                  (*Window size for the qualitative filter*)
```

Figure A.5: Parameter for qualitative representation generation.

A.3. Simulation

The input to simulation process is the qualitative representation of the bifurcation diagram, and the descriptor of an initial state. The qualitative representation is obtained from $GenerateQR[PI, PE]$ function. Figure A.6 illustrates the execution of this function; it shows the definition of the QR, the initial state, and the descriptor.

As observed in Figure A.6, all numerical values are discarded (observe *QS2*). Also if we observe *QS2*, we notice that two new landmarks are added to the Quantitative Space and no numerical values are needed. The simulation routine returns the History for the process as mention in description Algorithm 10.

```
QS2 = {{r0, r4, r1, r3, r2}, {x0, x1, x2, x3, x4}};
InitalState = {s1, r3};
Descriptor = {r4, "+", r3};
QR = {QS2, SEG, LINKS};
History = QualitativeSimulation[QR, Descriptor, InitalState];
GenerateRepresentation[History]

Follow[{s1, r3, r1, -, +}]

FallOff[{s1, s3, r1, +}]

Follow[{s3, r1, r4, -, -}]

Perturbation[{s3, s3, r4, +}]

Follow[{s3, r4, r3, +, +}]
```

Figure A.6: illustrating the simulation function.

A.4. Control Planning

The Planning algorithm defines the parameter changes (descriptor) to move the system from one initial to a final state. The function receive as input the QR, the initial state, the goal state, and the index value for the creation of new landamarks. Since the planning algorithm creates new lardmarks, this parameter indicates the staring parameter value. For example if $parameterStart = 10$ the new landmarks created in the process will start naming them at $r = 10$. The output of this routine is the Descriptor and the new QS (new landmarks were added in the process). Figure A.7 shows the execution of the planning routine.

```
In[65]:= parameterStart = 10;
         InitalState = {s1, r3};
         GoalState = {s3, r3};
         {Descriptor, NewQS} = FindPath[QR, InitalState, GoalState, parameterStart]
Out[68]= {{r10, r3}, {{r0, r4, r10, r1, r3, r2}, {x0, x1, x2, x3, x4}}}
```

Figure A.7: illustrating the planning routine.

A.5. Plotting Qualitative Bifurcation Diagrams

The function *GenerateQR* defines the behavior of the dynamical system when the parameter is changed. Changes are defined in a descriptor as explained before, and the result of the function is the *History*. To illustrate the *History* we use the function *PlotSimulation*. But first the system creates the arrows with the function *GraphHistory*. Note that the parameter QS has the real values of the landmarks. This is the only function that works with quantitative information. Example of running the function can be observed in Figure A.8.

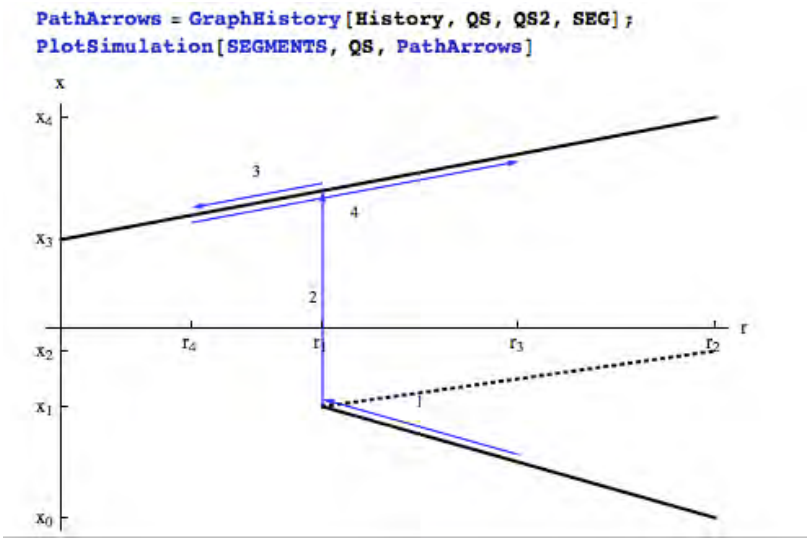


Figure A.8: Graphical output of a simulation. In this Figure we observe the path (blue arrows) resulting form the simulation algorithm.

A.6. Chapter Conclusions

In this chapter we described the use of QBD. This application is the implementation of the three algorithms described in this thesis. During this chapter the parameters that can be varied were described. Also the chapter illustrates how to execute the functions to operate the application.

References

[Barrera08] Barrera, J., Flores, J. J., and Fuerte-Esquivel, C. R. Plotting of complete bifurcation diagrams using a dynamic environment particle swarm optimization algorithm. *En IC-AI*, pages. 399–406. 2008.

[Boukas02] Boukas, E.-K. and Liu, Z.-K. *Deterministic and stochastic time delay systems*. Control engineering. Birkhäuser, Boston, Basel, Berlin, 2002. ISBN 3-7643-4245-5.
URL `http://opac.inria.fr/record=b1104718`

[Boyce01] Boyce, W. E. and DiPrima, R. C. *Elementary Differential Equations*. Wiley, 2001. ISBN 0-471-31998-8.

[Crawford91] Crawford, J. D. Introduction to bifurcation theory. *Reviews of Modern Physics*, 63(4), 1991.

[deKleer82] de Kleer, J. and Brown, J. S. Foundations of envisioning. *En Proc. of AAAI-82*, pages. 434–437. Pittsburgh, PA, 1982.

[deKleer84] de Kleer, J. and Brown, J. S. A qualitative physics based on confluences. *Artif. Intell.*, 24(1-3):7–83, 1984.

[Ermentrout03] Ermentrout, B., Mahajan, A., and Reviewer. Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students. *Applied Mechanics Reviews*, 56(4):B53, 2003. doi: 10.1115/1.1579454.
URL `http://dx.doi.org/10.1115/1.1579454`

[Farina11] Farina, L. and Rinaldi, S. *Positive Linear Systems: Theory and Applications*. John Wiley and Sons, 2011.

[Farkas81] Farkas, M. *Qualitative theory of differential equations. Vol. 2*. Colloquia mathematica Societatis János Bolyai. Amsterdam Oxford New York, N.Y. North-Holland, 1981. ISBN 0-444-86173-4.
URL `http://opac.inria.fr/record=b1089766`

[Flores06] Flores, J. J. and Proskurowski, A. Qualitative reasoning and bifurcations in dynamic systems. *En MICAI*, pages. 259–271. 2006.

[Flores10] Flores, J. J., López, R., and Barrera, J. Particle swarm optimization with gravitational interactions for multimodal and unimodal problems. *En MICAI (2)*, pages. 361–370. 2010.

[Forbus84] Forbus, K. D. Qualitative process theory. *Artif. Intell.*, 24(1-3):85–168, 1984.

[Forbus97] Forbus, K. D. Qualitative reasoning. *En The Computer Science and Engineering Handbook*, pages. 715–733. 1997.

[Ivancevic08] Ivancevic, V. G. and Ivancevic, T. T. *Complex Nonlinearity: Chaos, Phase Transitions, Topology Change and Path Integrals*. Understanding Complex Systems. Springer, Berlin, Heidelberg, 2008.

[Johan De Kleer84] Johan De Kleer, J. S. B. A qualitative physics based on confluences. *Artif. Intell.*, 24(1-3):7–83, 1984.

[Kay00] Kay, H., Rinner, B., and Kuipers, B. Semi-quantitative system identification. *Artif. Intell.*, 119(1-2):103–140, 2000.

[Kuipers94] Kuipers, B. *Qualitative reasoning - modeling and simulation with incomplete knowledge*. MIT Press, 1994. ISBN 978-0-262-11190-4.

[Lakshmanan96] Lakshmanan, M. and Murali, K. *LINEAR AND NONLINEAR OSCILLATORS*, cap. 2, pages. 5–19. 1996.

[Lee93] Lee, W. W. and Kuipers, B. A qualitative method to construct phase portraits. *En AAAI*, pages. 614–619. 1993.

[Lorenz63] Lorenz, E. N. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20:130–141, 1963.

[Mandelbrot04] Mandelbrot, B. B. *Fractals and chaos : the Mandelbrot set and beyond*. Springer, New York, Berlin, Paris, 2004. ISBN 0-387-20158-0.
URL `http://opac.inria.fr/record=b1117480`

[Morris81] Morris, C. and Lecar, H. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35(1):193–213, jul 1981.
URL `http://dx.doi.org/10.1016/s0006-3495(81)84782-0`

[Peitgen04] Peitgen, H.-O., Jürgens, H., and Saupe, D. *Chaos and fractals : new frontiers of science*. Springer, New York, 2004. ISBN 0-387-20229-3.
URL `http://opac.inria.fr/record=b1100638`

[Poincaré85] Poincaré, H. Sur l'équilibre d'une masse fluide animée d'un mouvement de rotation. *Comptes rendus hebdomadaires de l'Académie des sciences*, 100:1067–1070, 1885.

[Ruelle70] Ruelle, D. and Taken, F. On nature of turbulence. *Comun Math Phys*, 20:167–192, 1970.

[Russell64] Russell, J. L. Kepler's laws of planetary motion: 16091666. *The British Journal for the History of Science*, 2:1–24, 6 1964. ISSN 1474-001X. doi:10.1017/S0007087400001813.

[Sacks90] Sacks, E. A dynamic systems perspective on qualitative simulation. *Artificial Intelligence*, 42:349–362, 1990.

[Scowen93] Scowen, R. S. Extended BNF - A Generic Base Standard. *En Proceedings of the 1993 Software Engineering Standards Symposium (SESS'93)*. ago. 1993.
URL `http://www.cl.cam.ac.uk/~mgk25/iso-14977-paper.pdf`

[Strogatz94] Strogatz, S. H. *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in Nonlinearity)*. Studies in nonlinearity. Perseus Books Group, 1994. ISBN 9780738204536.

[Tsumoto06] Tsumoto, K., Kitajima, H., and Yoshinaga, T. Bifurcations in Morris-Lecar neuron model. *Neurocomputing*, 69(4-6):293–316, 2006.

[Wiggins90] Wiggins, S. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Nº 2 en Texts in Applied Mathematics. Springer-Verlag, New York, NY, 1990.

[Winfree80] Winfree, A. T. *The geometry of biological time*. Biomathematics. Springer Verlag, New York, 1980. ISBN 0-387-09373-7.
URL `http://opac.inria.fr/record=b1085595`

[Wolfram10] Wolfram, R. *Mathematica*. Wolfram Research, Inc., 2010. ISBN 0-521-64314-7.

[Yang02] Yang, S.-K., Chen, C.-L., and Yau, H.-T. Control of chaos in lorenz system. *Chaos, Solitons and Fractals*, 13(4):767 – 780, 2002. ISSN 0960-0779.

[Yip87] Yip, K. M. Extracting qualitative dynamics from numerical experiments. *En Proc. 7th National Conf. on Artificial Intelligence (AAAI-87)*, pages. 665–671. AAAI Press/The MIT Press, 1987.