# Computation of Invariant Manifolds Using AUTO-07p

Rui Chen

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

August 2007

*Your file Votre référence*
*ISBN: 978-0-494-34738-6*
*Our file Notre référence*
*ISBN: 978-0-494-34738-6*

## Abstract

Computation of Invariant Manifolds Using AUTO-07p

Rui Chen

There is much interest recently in the computation of invariant manifolds of vector fields. This thesis presents a continuation method for computing two-dimensional stable and unstable manifolds arising in dynamical systems.

The computations are illustrated using a well-known example, namely, the Lorenz system, for which detailed results are presented for the so-called "Lorenz manifold", *i.e.*, the two-dimensional stable manifold of the origin, as well as for the two-dimensional unstable manifold of one of the two symmetric nonzero equilibria. A number of the infinitely many intersection curves of these manifolds are also determined accurately. All computations are carried out using the numerical continuation software AUTO, specifically its most recent version, AUTO-07p.

Various diagrams are given to illustrate the numerical results. Software based on OpenGL and Glut has been developed to visualize the numerically computed manifolds, using a triangulation of the data computed with AUTO.

## Acknowledgments

I would like to express my sincere appreciation to my supervisor Professor, Eusebius J. Doedel, for his guidance, support, documents, and patience, for completing my thesis. Also, I am particularly thankful to Qiang Zheng and Chenghai Zhang, to whom I am very much indebted: this thesis would have never appeared without their help and encouragement.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, we first introduce briefly some basic terminology related to dynamical systems and manifolds, and then we present various existing numerical methods for computing stable or unstable manifolds. The outline of this thesis appears at the end of this chapter.

## 1.1 General view of dynamical systems

Dynamical systems are mathematical objects used to model physical phenomena whose state changes over time. These models are used in financial and economic forecasting, medical diagnosis, environmental modeling, and many other applications. The evolution of the system state refers to a sequence or continuous trajectory through the space of possible system states.

The space of possible system states is called the *state space* of the dynamical system, which is represented by the points of some set $U$, which is often a subset of $R^n$. One basic goal of research in dynamical systems is to characterize or determine the long-term behavior of the system.

Two kinds of dynamical systems are common, one is called a *continuous dynamical systems*, in which the time variable $t$ changes continuously. In this case the state is usually a function $\mathbf{x}(t)$ of time, where $\mathbf{x} \in U$. The change of state is often described by a differential equation, a simple form of which is:

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) \quad , \tag{1.1}$$

where $\mathbf{f}$ is a function of the state, and the map $\mathbf{f} : \Omega \mapsto R^n$ is sufficiently smooth. $\Omega$ is an open subset of $R^n$, and in general, the initial state is also specified. The famous Lorenz system [53] is of this kind and certain of its aspects will be studied in some detail in this thesis.

In contrast, we refer to a dynamical system as a *discrete dynamical system* when movements are observed in discrete time intervals. In this case, the evolution may take the form

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n) \quad ,$$

where $n$ is a integer, $\mathbf{x}_n$ represents the measurements at time $n$, and $\mathbf{f}(\mathbf{x})$ is a given function. The famous Logistic Equation [67] belongs to this category.

Even very simple dynamical systems, can give highly complicated behavior. Most of the time, it is impossible to give analytical solutions; instead, the qualitative behavior is studied. Numerical methods, with the help of modern computer technology, often help us understand the behavior of the solutions.

## 1.2 Differential equations and dynamical systems

Equation (1.1) represents a system of ordinary differential equations: these equations are used to define a continuous-time dynamical system. Since $t$ does not appear explicitly in $\mathbf{f}(\cdot)$, the system is called *autonomous.*

The right hand side of Equation (1.1), is also called *vector field*, since to each point $\mathbf{x}$ there is assigned a vector $\mathbf{f}(\mathbf{x})$, see [51].

A solution $\mathbf{x}_0$ satisfying

$$\mathbf{f}(\mathbf{x}_0) = 0 \quad ,$$

is called an *equilibrium point* of Equation (1.1). Once a system is at such a point $\mathbf{x}_0$ in the state space, it is at rest. Equilibrium solutions are constant solutions; sometimes we also use the term *stationary solutions* for them. The study of these solutions is very important in dynamical systems.

## 1.3 Stable and unstable manifolds

Many nonlinear phenomena can be explained by understanding the behavior of objects present in the dynamics. Important objects for this purpose are *invariant manifolds.* They are embedded in a phase space, and are invariant under the flow, *i.e.*, orbits that start out in the manifold remain in it. Manifolds have dimensions, for example, one-dimensional manifolds may include a line, a circle, and two-dimensional manifolds look like a disk, a plane, the surface of a sphere, and so on; see [76].

In particular, we will be interested in invariant surfaces known as *stable and unstable manifolds* of equilibria. Understanding the structure of these manifolds is of considerable significance; it helps us to understand the eventual dynamics of systems as time evolves.

Stable and unstable manifolds are the points sets in state space that tend toward a special equilibrium point, namely, a *saddle*, in forward or backward time under evolution of the system. Saddle points are very important equilibria. We will describe saddle points in more detail in Chapter 2.

For Equation (1.1), suppose $\mathbf{f}(\mathbf{x}_0) = 0$ and the Jacobian $D\mathbf{f}(\mathbf{x}_0)$ of $\mathbf{f}$ at $\mathbf{x}_0$ has $m$ eigenvalues with positive real parts and $(n - m)$ eigenvalues with negative real parts. From the Stable and Unstable Manifold Theorem [37], one knows that a local unstable manifold $W^u_{loc}(\mathbf{x}_0)$ exists in a neighborhood of $\mathbf{x}_0$. The global unstable manifold $W^u(\mathbf{x}_0)$ may be defined as the globalization of $W^u_{loc}(\mathbf{x}_0)$ under the flow. $W^u(\mathbf{x}_0)$ is a $m$-dimensional manifold.

It is well known how to compute global one-dimensional unstable (or stable) manifolds of an equilibrium of a smooth vector field, since only two trajectories define such a manifold. Such manifolds can be generated using an initial point close to the equilibrium. The computation error stays bounded due to the attraction. This thesis, however, focuses on how two-dimensional stable or unstable manifolds can be computed very effectively using the software package AUTO, [23, 26]. Furthermore, we need only consider the case of an unstable manifold, because a stable manifold can be treated in the same way, after time is reversed.

## 1.4 Numerical methods for computing stable or unstable manifolds

For most dynamical systems, the only general way of studying their stable and unstable manifolds is by computing them numerically, as they usually cannot be found analytically. Consequently, computational scientists have developed algorithms for computing stable and unstable manifolds for vector fields and for discrete maps.

One of the basic methods for vector fields is numerical integration, which is suited for computing a one-dimensional unstable manifold of an equilibrium of a vector field. But it is a challenge to compute a global unstable manifold of dimension at least two, since simple numerical integration of the flow is not sufficient when higher dimensional manifolds are desired: see [8] for detailed discussion.

Another basic numerical method is *continuation*: for the case of equilibria, its main idea is to start with locating at least one equilibrium at certain fixed parameter values, and then by varying one of the system's parameters, one continues the obtained equilibrium with respect to this parameter. Some special points may be detected during this process. In this thesis it will be shown that numerical continuation is also an extremely effective method for computing $2D$ stable and unstable manifolds.

There are several software packages for the numerical analysis of dynamical systems. Most are for the computation of one-dimensional solution manifolds. AUTO is one of the earliest packages and perhaps the most widely used. We show in Chapter 6 how the continuation methods in AUTO can be used to compute two-dimensional stable and unstable manifolds in the Lorenz system.

## 1.5 Different approaches

For computing an unstable manifold, we can take a small $(m-1)$ sphere $S_\delta \subset W^u_{loc}(\mathbf{x}_0)$ with radius $\delta$ around $\mathbf{x}_0$, and evolve $S_\delta$ under the flow to generate the manifold $W^u(\mathbf{x}_0)$. In this way, the computation of the one-dimensional unstable manifold of an equilibrium of a vector field is straightforward, because the process is just to evolve two points at distance $\delta$ from $x_0$ under the flow.

This method is not suited for computing a two or more dimensional unstable manifold. The reason is that $S_\delta$ is generally discretized by some mesh, and any mesh on $S_\delta$ will deteriorate very rapidly, so that it will not be a good representation of $W^u(\mathbf{x}_0)$ as an $m-$dimensional manifold. Different approaches are designed to solve this problem. An often used idea is to grow $W^u(\mathbf{x}_0)$ from a local neighborhood of $\mathbf{x}_0$. The methods differ in how a good mesh representation of $W^u(\mathbf{x}_0)$ is computed during this process. We briefly introduce some of the techniques below.

Most algorithms that compute two-dimensional unstable manifolds are for vector fields. Guckenheimer and Worfolk describe a method to compute the two-dimensional unstable manifold of the origin for the Lorenz system. It starts with a small circle $S_\delta$ around $x_0$ in the unstable eigenspace. Then, by iterating, one obtains a family of circles as an approximation of the unstable manifold; see [39].

Krauskopf and Osinga [48, 49] compute $W^u(x_0)$ as a sequence of geodesic circles. This method computes new mesh points on the next geodesic circle by solving appropriate boundary value problems. The manifold is "grown" as a sequence of discretized geodesic circles until they are no longer smooth circles. Adaptive coordinate systems are needed to trace the manifold.

Doedel [50] computed unstable manifolds by following entire orbits that lie on the manifold by numerical continuation. The procedure is stepwise and each step is a two point boundary value problem. This method is very accurate and flexible because different boundary and integral conditions can be specified. This method is the one that we shall focus on in this thesis.

In the method of Guckenheimer and Johnson, the manifold is computed as a set of curves. The length of these curves grows quickly and interpolations are used to place new points on the curves to generate the unstable manifold adequately. But in places where there are sharp folds, it becomes difficult to carry out this interpolation technique. For further algorithms, one can consult [43].

There are also several methods for the computation of one or two-dimensional unstable manifolds of maps, with different algorithms being applied to different applications: one can find further information in [20, 33, 65].

## 1.6 Organization of the thesis

In this thesis, we demonstrate the performance of numerical methods for invariant manifolds, based on continuation methods and in a boundary value problem setting, by computing stable and unstable manifolds of the well-known Lorenz system [53]. We use AUTO to carry out these calculations. To better understand the structure of the manifolds, a new visualization tool is developed, with which we can view manifolds in a 3D graphics environment. This tool will be useful to researchers who need to do similar manifold calculations, and in other applications where families of orbits need to be displayed as surfaces.

In this thesis, the software package AUTO-07p is used to compute manifolds, which combines almost all the facilities of AUTO97 [26] and AUTO2000 [27].

The outline of this thesis is as follows. The Lorenz equations are discussed in Chapter 2, where we also introduce some of the mathematical concepts and notation that we use throughout the thesis. In Chapter 3 basic integration methods for initial value problems are described, followed by a detailed presentation of the application of basic integration methods to the Lorenz system. Chapter 4 discusses collocation methods for boundary value problems, while Chapter 5 introduces continuation methods. The detailed computation of stable and unstable manifolds of the Lorenz system using AUTO is presented in Chapter 6, packages related to AUTO are briefly introduced. Chapter 7 introduces the computation of heteroclinic connections. The design and development of a new AUTO data visualization tool, using a triangulation technique, is explained in Chapter 8. Chapter 9 presents conclusions and discusses some topics that merit future work. Appendices explain the implementation used in AUTO for the Lorenz system and a brief description of the AUTO utilities is given. Finally, packages related to AUTO are introduced.

Chapter 2

# Mathematical Description and Background

In this chapter we introduce some mathematical concepts in dynamical systems, as well as some notation that we use in this thesis. We also give the history of the Lorenz system, and we briefly describe it in a mathematical context.

## 2.1 Stability of an equilibrium point

In order to study the evolution of a dynamical system, it is essential to know whether trajectories that start close to each other display a similar qualitative behavior. For this purpose, we need to explain some basic concepts related to points in the state space of a dynamical system.

An equilibrium is considered *stable* if the system always returns to it after small disturbances. If the system moves away from the equilibrium after small disturbances, then the equilibrium is *unstable*.

More precisely, consider equation $\mathbf{x}' = \mathbf{f}(\mathbf{x})$. Assume that $\mathbf{x}_0$ is an equilibrium point, so $\mathbf{f}(\mathbf{x}_0) = 0$. Consider now a solution in a neighborhood of $\mathbf{x}_0$. The equilibrium point $\mathbf{x}_0$ has the following classifications:

- It is called *stable* if, for each $\varepsilon > \mathbf{0}$ there is a $\delta > 0$ so that whenever $|\mathbf{x} - \mathbf{x}_0| < \delta$, then $|\mathbf{x}(t) - \mathbf{x}_0| < \varepsilon$ for all $t > 0$;

- It is called *unstable* if it is not stable;
- It is called *asymptotically stable* if it is stable and there is some $\delta > 0$ so that $\mathbf{x}(t) \to \mathbf{x}_0$ when $t \to \infty$ for all $\mathbf{x}(0)$ with $|\mathbf{x}(0) - \mathbf{x}_0| < \delta$.

For simplicity, we will only consider asymptotic stability in this thesis.

## 2.2 Bifurcation theory

Many dynamical systems depend on parameters, *e.g.*, the Lorenz system, fluid flow problems, heat diffusion problems, *etc.* They have the general form

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), \lambda) \quad , \quad \mathbf{x} \in R^n \quad , \tag{2.1}$$

where $\lambda$ is a parameter in $R$. There exists a large number of problems for which the number of solutions can change abruptly and the structure of solutions can change dramatically when a parameter passes through some critical values. This kind of phenomenon is called a *bifurcation*, and the parameter value at which the critical value occurs is called a *bifurcation point*. One can consider bifurcation theory as a method for studying how solutions of a nonlinear problem and their stability change as parameters vary: see [51].

Now suppose $\mathbf{x}_0$ is an equilibrium solution of Equation (2.1), in another words, $\mathbf{f}(\mathbf{x}_0, \lambda) = 0$. If $\mathbf{x}_0$ is an asymptotically stable equilibrium, then, as $\lambda$ varies, its stability might, in the simplest instances, change in two ways (see Figure 2.1 and Figure 2.2 for graphical representation; see [32]):

1. A real eigenvalue of $\mathbf{f}'(\mathbf{x}_0, \lambda)$ crosses the imaginary axis. This is called a *stationary bifurcation*;

2. A pair of complex conjugate eigenvalues of $\mathbf{f}'(\mathbf{x}_0, \lambda)$ cross the imaginary axis. This is the so-called *Hopf bifurcation*, and, typically, periodic solutions appear in this case.

A bifurcation diagram depicts the transition between different types of behavior as a parameter of the system is varied. It plots a system parameter on the horizontal axis and a representation of an attractor, usually $\| \mathbf{x} \|$, ($\| \cdot \|$ denotes a norm) or other measure of $\mathbf{x}$ on the vertical axis.

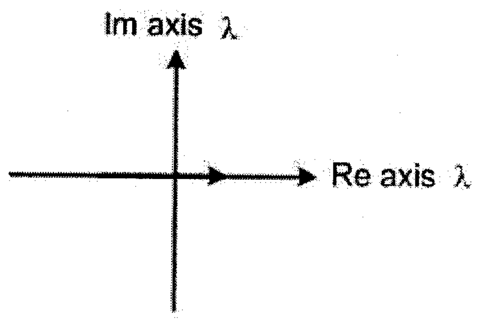


Figure 2.1: Stationary bifurcation

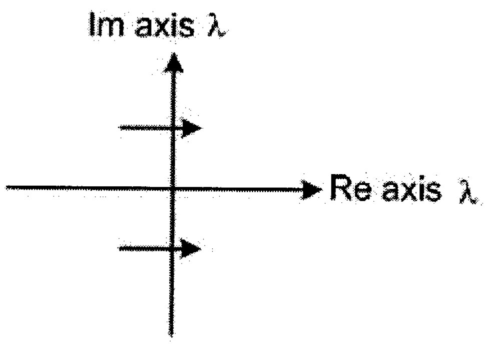


Figure 2.2: Hopf bifurcation

There are a number of computer tools for studying bifurcations, such as AUTO, DsTool, and a more recent implementations, such as `CONTENT`, `MATCONT`, `Oscill8`, *etc.*, see Appendix C for more details.

## 2.3 Phase portraits

*Phase portraits* are an invaluable tool in studying dynamical systems. They consist of a plot of typical trajectories. The axes of a phase portrait graph correspond to state variables. More simply, one can think that the phase portrait of a dynamical system actually partitions the state space into orbits; it can be interpreted as an image that looks like the flow of a liquid. See [51] for the formal definition.

The study of the phase diagram is an indispensable tool in understanding the qualitative behavior of solutions to a system. One can determine the number and types of the possible states by observing the phase portrait. The software package Maple [54], for example, can be used to draw phase diagrams, and so can other software. Although it is not possible to draw all orbits, the crucial orbits can be displayed in the phase portrait of a dynamical system.

In a differential equation such as the Lorenz system, the phase portrait represents the way a 3-dimensional system, *i.e.*, a system with three degrees of freedom, will evolve over time. Each point in the phase portrait corresponds to a possible state of the system. The curves are trajectories, which show how the three variables will evolve given a set of initial conditions.

As another example, consider a $2D$ Predator-Prey model

$$\begin{cases} x_1' = 3x_1(1-x_1) - x_1x_2 - p_1(1-e^{-5x_1}) , \\ x_2' = -x_2 + 3x_1x_2 . \end{cases}$$

where $x_1$ represents "fish" and $x_2$ represents "sharks", and the term $p_1\ (1 - e^{-5x_1})$ , is "fishing" with "fishing-quota" $p_1$.

Computation of stationary solutions can be easily done by AUTO; we can then observe how solutions change when $p_1$ varies. Figure 2.3 is such bifurcation diagram showing how $x_1$ changes when $p_1$ varies.

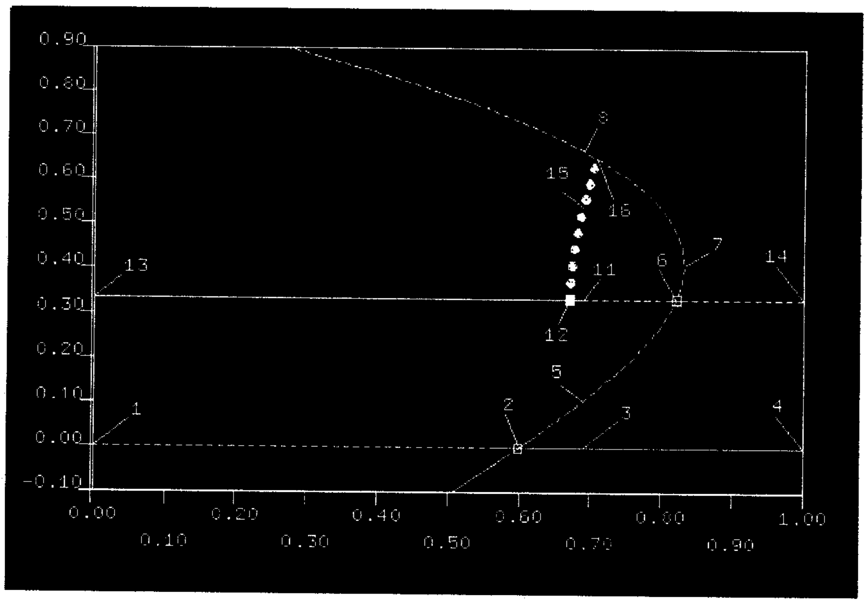


Figure 2.3: Bifurcation diagram of a Predator-Prey model

The periodic solution family is represented by the solid dots. Solid/dashed lines denote stable/unstable stationary solution. Open squares are *branch points* or *bifurcation points*; the solid square is a Hopf bifurcation [24, 25]. (See also Section 2.2).

We can also observe how orbits evolve when $p_1$ is at some particular value. For instance, when $p_1 = 0.69$, there are five labeled solutions, namely, solution $3, 5, 8, 11, 15$, where solution "15" is a periodic solution. Below we sketch the phase portrait when $p_1 = 0.708$.

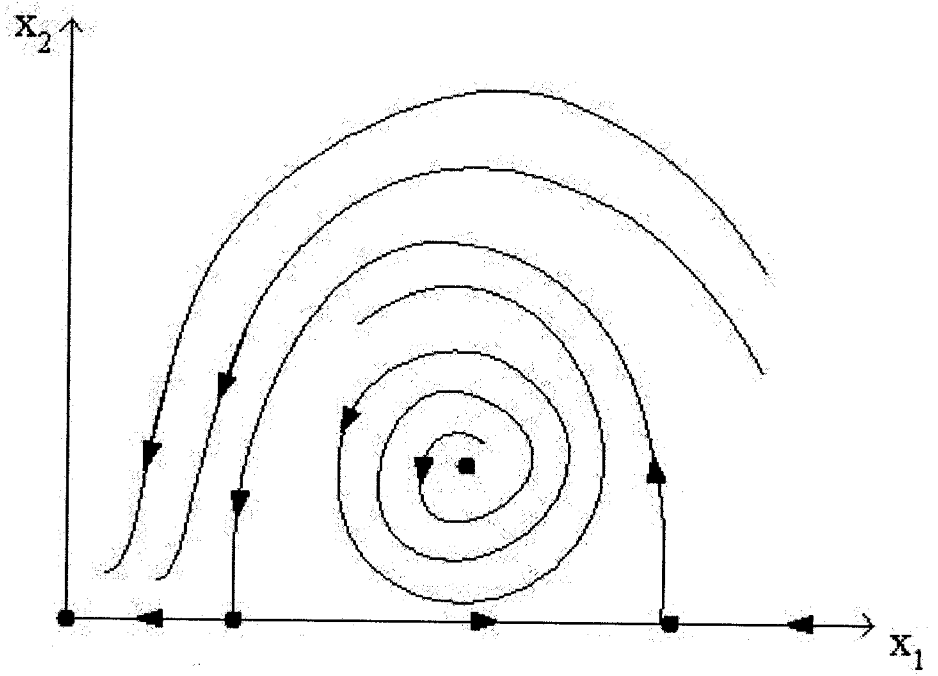


Figure 2.4: Sketch of phase portrait of the Predator-Prey model at $\lambda = 0.708$ where there is a heteroclinic cycle

In many applications the dimension of the state space is very large. For example, in theoretical neuroscience, a very high dimension is possible, *e.g.*, one dimension for each neuron, to represent possible states of the brain: see [81]. Any point in this high-dimensional space will correspond to one particular state of the system. A curve in this space will correspond to a particular process, a stream of activity over time. Some methods have been developed for this purpose: The key point is to project high dimensional spaces onto two or three dimensions. A particular high-dimensional visualization tool is "HiSee" [42].

## 2.4 Linearization technique

The solutions of linear systems can, in principle, be found explicitly. However, real life problems typically are modeled by nonlinear systems. In this case, we can usually only qualitatively describe the solutions. What happens around an equilibrium point can often be described by a procedure called *linearization*. In such a case, we hope that the behavior of the solutions of the linear system will locally be similar to that of the nonlinear system.

Below is a simple illustration of the linearization technique. Consider the autonomous system (1.1). If $\mathbf{x} = (x_1, x_2)^T$, we can rewrite the system as

$$\begin{cases} x_1' &= f(x_1, x_2) \quad , \\ x_2' &= g(x_1, x_2) \quad , \end{cases} \tag{2.2}$$

where $x_1, x_2 \in R$, $f, g : R^2 \mapsto R$. Assume that $(x_1^0, x_2^0)$ is an equilibrium point. Compute the partial derivatives:

$$\frac{\partial f}{\partial x_1}(x_1^0, x_2^0), \quad \frac{\partial f}{\partial x_2}(x_1^0, x_2^0), \quad \frac{\partial g}{\partial x_1}(x_1^0, x_2^0), \quad \text{and} \quad \frac{\partial g}{\partial x_2}(x_1^0, x_2^0), \tag{2.3}$$

and write down the matrix

$$\begin{pmatrix} \frac{\partial f}{\partial x_1}(x_1^0, x_2^0) & \frac{\partial f}{\partial x_2}(x_1^0, x_2^0) \\ \frac{\partial g}{\partial x_1}(x_1^0, x_2^0) & \frac{\partial g}{\partial x_2}(x_1^0, x_2^0) \end{pmatrix} \quad . \tag{2.4}$$

This matrix is the *Jacobian* matrix, and we usually represent it as $\mathbf{f}'(\mathbf{x}_0)$, or $D\mathbf{f}(\mathbf{x}_0)$ in short, where $\mathbf{x}_0 = (x_1^0, x_2^0)^T$ in this case.

In the general case, the eigenvalues $\lambda_d$ (where $d = 1, 2, \ldots n$) are the roots of the characteristic polynomial $det(D\mathbf{f}(\mathbf{x}_0) - \lambda I) = 0$. Finding the eigenvalues of the Jacobian is crucial, because we can deduce the fate of the solutions around the equilibrium point from the eigenvalues.

There are three basic types of equilibria in $R^2$, their names are inspired by the flow near the equilibrium point; see Table 2.1:

| Equilibrium type | Characteristics of eigenvalues |
|---|---|
| *Node* | two real eigenvalues of the same sign |
| *Spiral* | A pair of conjugate complex eigenvalues |
| *Saddle* | Real eigenvalues of different sign |

Table 2.1: Classification of an equilibrium point

The graphical representation of a node and a spiral is shown below; see [32].

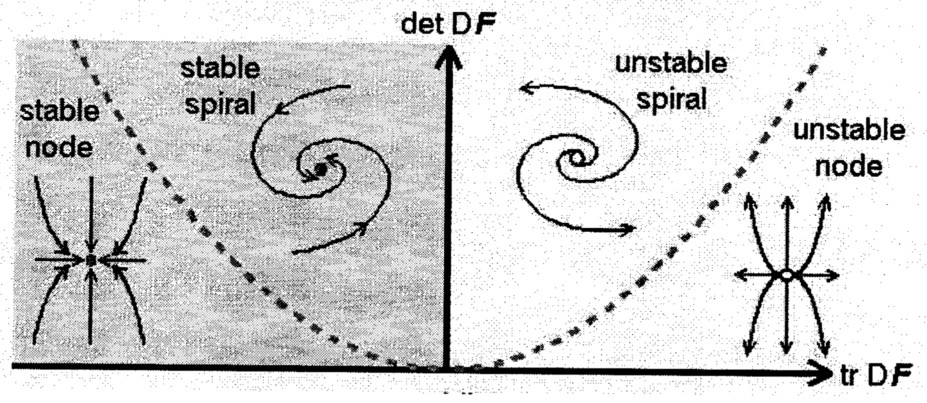


Figure 2.5: Vector field flow around equilibria

## 2.5 Definition of stable and unstable manifolds

From the above discussion, we know that a saddle point is especially interesting since, in the case of $R^2$, there are four special orbits associated with a saddle point. There will be two solutions which approach the equilibrium point as $t \rightarrow \infty$, and two more solutions which approach the equilibrium point as $t \rightarrow -\infty$. Each of these four solutions is called a *separatrix.*

In general, a saddle is an equilibrium point where at least one eigenvalue has a positive real part, and at least one eigenvalue has a negative real part. The points in the phase space which approach the saddle for $t \rightarrow \infty$ form the *stable manifold.* The eigenspace for the eigenvalues with negative real part is tangential to the stable manifold: see [50]. The *unstable manifold* consists of points approaching the saddle for $t \rightarrow -\infty$. Saddles and their stable manifolds and unstable manifolds are usually the boundaries of a basin of attraction. Figure 2.6 illustrates how the flow looks like around a saddle in $R^2$.

The stable and unstable manifolds $W^s(\mathbf{x}_0)$ and $W^u(\mathbf{x}_0)$ of a saddle equilibrium $\mathbf{x}_0$ in

$R^n$ are formally defined as

$$W^s(\mathbf{x}_0) : \; = \; \{\mathbf{x} \in R^n | \lim_{t \to \infty} \phi^t(\mathbf{x}) = \mathbf{x}_0\} \quad , \tag{2.5}$$

$$W^u(\mathbf{x}_0) : \; = \; \{\mathbf{x} \in R^n | \lim_{t \to \infty} \phi^{-t}(\mathbf{x}) = \mathbf{x}_0\} \quad , \tag{2.6}$$

respectively, where $\phi^t$ is the flow of the system, that is, how a set of trajectories evolve from some part of the plane. Trajectories on the stable (or unstable) manifold converge to $\mathbf{x}_0$ in forward (or backward) time. Knowledge of these manifolds is crucial, as they organize the dynamics on a global scale. Figure 2.7 sketches a possible scenario of the dynamics organized by the stable and unstable manifolds.

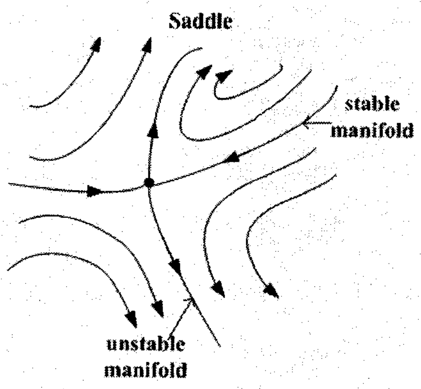


Figure 2.6: Vector field flow around a saddle point

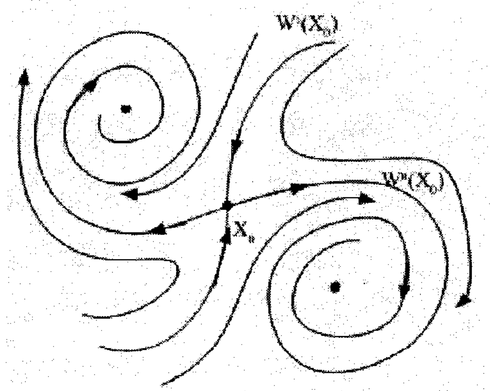


Figure 2.7: Illustration of stable and unstable manifolds

## 2.6 The Lorenz system

### 2.6.1 The history of the Lorenz system

The history of the Lorenz system can be traced back to the early 1960s. A meteorologist, Edward Lorenz, was trying to model the Earth's atmosphere, that is, to model the convective motion of a volume of air "which is warmed from below and cooled from above" [63]. His final model involved only three rather elementary equations:

$$\begin{cases} x_1' &= \sigma(x_2 - x_1) \quad , \\ x_2' &= \rho x_1 - x_2 - x_1 x_3 \quad , \\ x_3' &= x_1 x_2 - \beta x_3 \quad , \end{cases} \tag{2.7}$$

where $\mathbf{x} = (x_1, x_2, x_3) \in R^3$, $\sigma$, $\rho$ and $\beta$ are physical constants; $x_1$ is proportional to the intensity of the convective motion, and $x_2$ is proportional to the temperature difference between the ascending and descending currents. The third component, $x_3$, is proportional to the distortion of vertical temperature profile from linearity. Another interesting aspect is that the Lorenz system (2.7) has the symmetry $(x_1, x_2, x_3) \mapsto (-x_1, -x_2, x_3)$ of rotation by $\pi$ radians about the $x_3-$axis, *i.e.*, if $(x_1(t), x_2(t), x_3(t))$ is a solution, then so is $(-x_1(t), -x_2(t), x_3(t))$.

### 2.6.2 Parameterization

The system parameters of the Lorenz system are $\rho$, $\sigma$ and $\beta$. The most important parameter is $\rho$, also referred to as the *Rayleigh number*. In the context of convection in a fluid or the atmosphere, $\rho$ is proportional to the temperature difference across the layer, which is responsible for driving the fluid motion at a rate given by the variable $x_1$. The parameter $\sigma$ is called the *Prandtl number*, while $\beta$ is a positive parameter.

Usually $\sigma$ and $\beta$ are set to the values 10 and 8/3, respectively. The "standard" value of $\rho$ is 28. For these values, when one plots $x_3(t)$ against $x_1(t)$ as $t$ varies, the famous "Lorenz butterfly" appears.

### 2.6.3 Attractor

Different starting values of the three components lead to different behavior. However the dynamics evolves towards the same butterfly structure, which is therefore called an attractor. The dynamics on the attractor [38] will be similar in outline, but much different in details depending on the initial conditions, *i.e.*, chaotic [61]. The attractor is a so-called *fractal.* - "a geometric pattern that iterates infinitely often with recurring self-similarity" (see [15]); also called a *strange attractor*, in which the system exhibits chaotic behavior; see Figure 2.8 for a fragment of the attactor. We can also plot coordinates as a function of time; Figure 2.9 is a corresponding graph which shows $x_1$ component as a function of $t$.

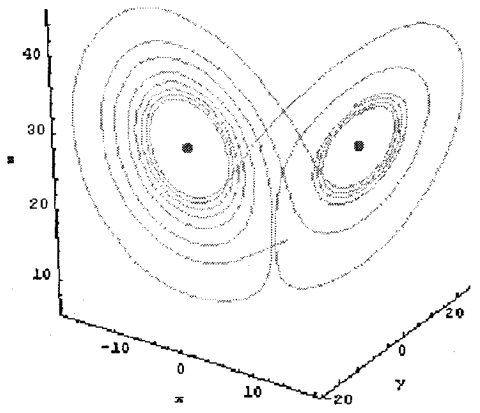


Figure 2.8: A fragment of the Lorenz attractor

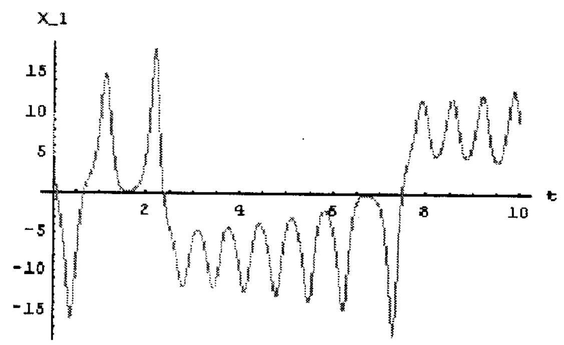


Figure 2.9: The $x$−coordinate as a function of time

We can see orbits go back and forth in an irregular manner between the two "wings" of the "butterfly". A important property of chaos is the sensitive dependence on initial conditions, or poetically, "the butterfly effect". We will look at the Lorenz system in more detail in the following section.

### 2.6.4 Eigenvectors and eigenvalues of the Lorenz system

The Lorenz system is a classic example of a vector field with a chaotic attractor. The computation of its two-dimensional stable manifolds, known as the Lorenz manifold, is useful to exhibit the interesting behavior of the system. Different algorithms have been developed: we will focus on making use of AUTO in the following sections.

The origin $\mathbf{0}^T = (0,0,0)$ is a stationary solution of Equation (2.7). The Jacobian matrix of $f$ is

$$D\mathbf{f} = \begin{bmatrix} -\sigma & \sigma & 0 \\ -x_3+\rho & -1 & -x_1 \\ x_2 & x_1 & -\beta \end{bmatrix} .$$

Evaluating the Jacobian matrix at 0 we have

$$D\mathbf{f}(0) = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho & -1 & 0 \\ 0 & 0 & -\beta \end{bmatrix} ,$$

so the characteristic polynomial of $\mathbf{f}$ at the origin is

$$\lambda^3 + (1+\sigma+\beta)\lambda^2 + (\beta\sigma+\beta+\sigma-\rho\sigma)\lambda + \beta\sigma(1-\rho) = 0 \quad ,$$

from which

$$(\lambda+\beta)[\lambda^2 + (\sigma+1)\lambda + \sigma(1-\rho)] = 0 \quad .$$

The eigenvalues are

$$\begin{aligned}
\lambda_1 &= -\beta \quad , \\
\lambda_{2,3} &= \frac{-(\sigma+1) \pm \sqrt{(\sigma+1)^2 - 4\sigma(1-\rho)}}{2} \quad .
\end{aligned}$$

If $\rho < 1$, all three eigenvalues are negative real numbers. If $\rho > 1$, then $\lambda_2$ becomes positive, so the stability of origin changes from attracting to repelling. For instance, if we fix the parameters at the standard values $\rho = 28, \sigma = 10$ and $\beta = 8/3$, we will obtain one unstable eigenvalue $\lambda_2^u \approx 11.828$ and two stable eigenvalues $\lambda_1^s \approx -2.667$ and $\lambda_3^s \approx -22.828$. The eigenvectors that correspond the two stable eigenvalues span the stable eigenspace $E^s(\mathbf{0})$, and the Lorenz manifold $W^s(\mathbf{0})$ is tangent at $\mathbf{0}$ to the eigenspace $E^s(\mathbf{0})$.

There are two more stationary solutions of Equation (2.7) when $\rho > 1$, namely,

$$\mathbf{x}_{1,2}^0 = (\pm\sqrt{\beta(\rho-1)}, \pm\sqrt{\beta(\rho-1)}, \rho - 1) \quad ,$$

approximately at $(\pm 8.485, \pm 8.485, 27)$ when $\rho = 28$, and symmetric to each other. Consider the Jacobian matrix at $\mathbf{x}_1^0$

$$D\mathbf{f}(\mathbf{x}_1^0) = \begin{bmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & -\sqrt{\beta(\rho-1)} \\ \sqrt{\beta(\rho-1)} & \sqrt{\beta(\rho-1)} & -\beta \end{bmatrix} \quad ;$$

its characteristic polynomial is

$$\lambda^3 + (1 + \sigma + \beta)\lambda^2 + (\beta\sigma + \beta\rho)\lambda + 2\beta\sigma(\rho - 1) = 0 \quad .$$

We find that each of these two equilibria has one negative eigenvalue, which is stable, and a pair of complex conjugate eigenvalues with positive real part, which is unstable. In fact, there exists a critical value of $\rho$, where the stability of these equilibria changes from stable to unstable for $\rho$ beyond that value. We will show how this value is detected in the following chapter. For more details, see [16, 50].

# Chapter 3

# Numerical Integration for Initial Value Problems

## 3.1 Initial value problem formulation

We know that a differential equation is an equation involving an unknown function and one or more of its derivatives. The equation is an ordinary differential equation (ODE) if the unknown function depends on only one independent variable. Problems involving ODEs can usually be reduced to the study of systems of first-order differential equations. For example, the second-order equation

$$\begin{cases} \mathbf{x}'' &= \mathbf{f}(\mathbf{x}(t), \mathbf{x}'(t)) \quad , \\ \mathbf{x}(0) &= \mathbf{x}_0, \quad \mathbf{x}'(0) = \mathbf{v}_0 \quad , \end{cases}$$

where $t \geq 0, \quad \mathbf{x}, \mathbf{f}(\cdot,\cdot,\cdot) \in \mathbf{R}^\mathbf{n}$, can be rewritten as two first-order equations

$$\begin{cases} \mathbf{x}'(t) &= \mathbf{v}(t) \quad , \\ \mathbf{v}'(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{v}(t)) \quad , \\ \mathbf{x}(0) &= \mathbf{x}_0, \quad \mathbf{v}(0) = \mathbf{v}_0 \quad . \end{cases}$$

With a differential equation we can associate initial conditions or boundary conditions, which are auxiliary conditions on the unknown function and its derivatives. If these conditions are specified at a single value of the independent variable, they are referred to as *initial conditions* and the combination of the differential equation and an appropriate number of initial conditions is called an *initial value problem, IVP* in short.

A initial value problem has the general form

$$\begin{aligned} \mathbf{x}'(t) &= \mathbf{f}(\mathbf{x}(t)) \quad , \quad t \geq 0 \quad , \quad \mathbf{x}, \mathbf{f}(\cdot,\cdot) \in \mathbf{R^n} \quad , \\ \mathbf{x}(0) &= \mathbf{x}_0 \quad . \end{aligned} \tag{3.1}$$

Note that we only consider *autonomous* equations here, where $\mathbf{f}$ does not explicitly depend on $t$. For instance, the Lorenz system (2.7) is an example of an autonomous initial value problem, see Section 2.6 for details.

There are two main ways of finding solutions of differential equations, namely, analytical methods and numerical methods. The former produce, when possible, exact analytical solutions in the form of general mathematical expressions. Numerical methods on the other hand, produce approximate solutions in the form of approximate values at discrete moments of time.

In general, since most solutions can not be found analytically, we must get an approximation to the solution. If we want to approximate the solution of a initial value problem, say, Equation (3.1), then numerical integration is the main tool.

## 3.2 Numerical integration of ordinary differential equations

There are many numerical integration methods available, for example, *Euler's method, BDF methods* (Backward Differentiation Formulas), *Runge Kutta methods, etc.*, see [77]. Below we briefly describe some of these.

### 3.2.1 Euler's method

The formula for the Euler method for solving Equation (3.1) is

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \ \mathbf{f}(\mathbf{x}_n) \quad , \quad n = 0, 1, 2, \cdots \quad . \tag{3.2}$$

The formula is not symmetric: It advances the solution through an interval $\Delta t$, but uses derivative information only at the beginning of that interval. It is explicit and only involves one step; the step error is $O(\Delta t^2)$, and the "global error" is $O(\Delta t)$.

There are several reasons that Euler's method is not recommended for practical use; among them the following two are most important:

- the method is not very accurate compared to other methods using the same stepsize,
- it is not very stable for stiff equations.

A representation of the local error of Euler's method is shown in Figure 3.1.

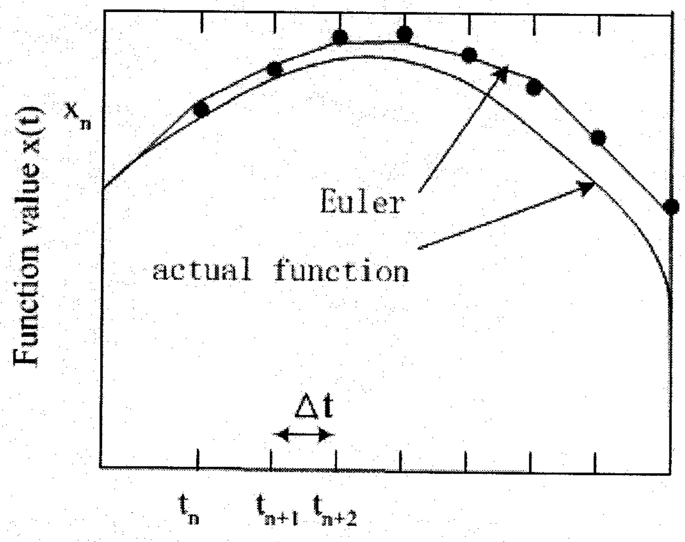


Figure 3.1: Representation of the error of Euler's method

### 3.2.2 The Runge-Kutta method

One of the most often used methods is the classical fourth-order Runge-Kutta formula, which has a certain elegance of representation:

$$\begin{aligned}\mathbf{k}_1 &= \Delta t\ \mathbf{f}(t_n\ , \mathbf{x}_n) \quad , \\ \mathbf{k}_2 &= \Delta t\ \mathbf{f}(t_n + \frac{\Delta t}{2}\ , \ \mathbf{x}_n + \frac{1}{2}\ \mathbf{k}_1) \quad ,\end{aligned}$$

$$\begin{aligned}
\mathbf{k}_3 &= \Delta t\ \mathbf{f}(t_n + \frac{\Delta t}{2}\ ,\ \mathbf{x}_n + \frac{1}{2}\ \mathbf{k}_2)\ \ , \\
\mathbf{k}_4 &= \Delta t\ \mathbf{f}(t_n + \Delta t\ ,\ \mathbf{x}_n + \mathbf{k}_3)\ \ , \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{1}{6}\ \mathbf{k}_1 + \frac{1}{3}\ \mathbf{k}_2 + \frac{1}{3}\ \mathbf{k}_3 + \frac{1}{6}\ \mathbf{k}_4\ \ . \qquad (3.3)
\end{aligned}$$

The fourth-order Runge-Kutta method requires four evaluations of the right hand side per step $t$: once at the initial point, twice at the midpoint, and once at the endpoint. From these derivatives the final function value (shown as a filled dot; see Figure 3.2 ) is calculated. This method achieves much better accuracy than Euler's method: the global error is $O(\Delta t^4)$.

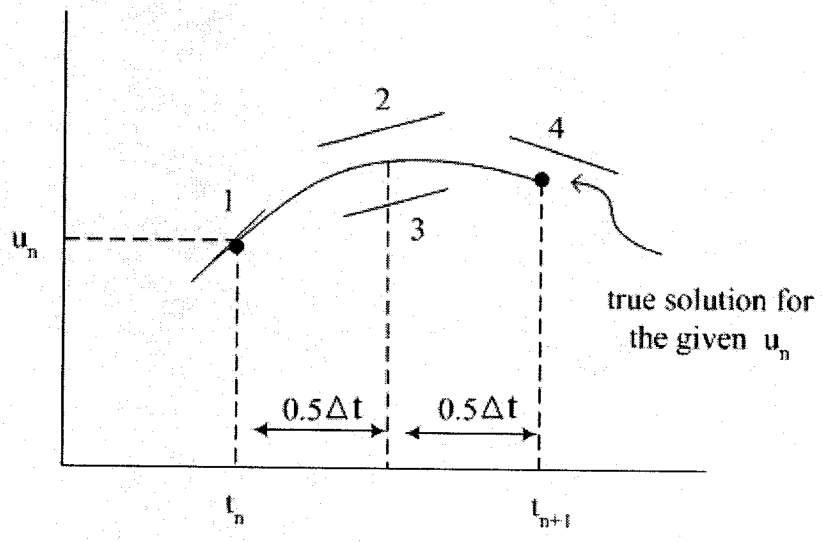


Figure 3.2: The fourth-order Runge-Kutta method

Actually, the Runge-Kutta method treats every step in a sequence of steps in identical manner. Prior behavior of a solution is not used in its propagation. This is mathematically correct, since any point along the trajectory of an ODE can serve as an initial point. The fact that all steps are treated identically also makes it easy to adopt this method to different applications. It is a competitive method, but unsuitable for very stiff equations; see [2] for more details .

## 3.3 Application of integration to the Lorenz system

In this section, we use the explicit 4th-order Runge-Kutta method to compute solutions to the Lorenz equations.

Suppose we take the initial conditions,

$$x_1(0) = 0.1 \quad , \quad x_2(0) = 0.2 \quad , \quad x_3(0) = 0.3 \quad ,$$

set $\sigma = 10$, $\beta = 8/3$, and let $\rho$ vary. Computing the solutions and plotting the phase space and time series, we can see that when $\rho < 1$, the solution decays rapidly to the origin, see Figure 3.3.

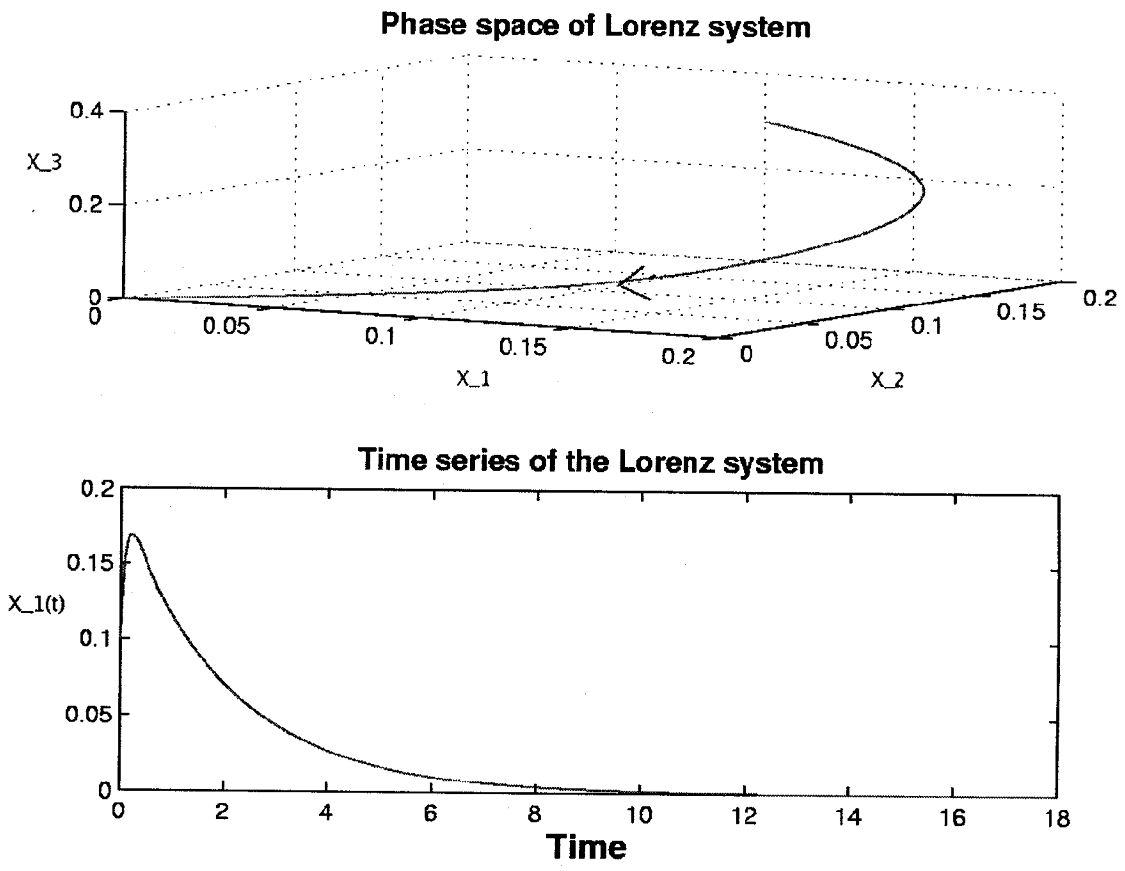


Figure 3.3: The Lorenz model: $\rho = 0.5$, step size= 0.03, # of steps= 600

As discussed in Section 2.6.4 of Chapter 2, we know that, for the given values of $\beta$ and $\sigma$, there is a critical value $\rho_c$ (a Hopf bifurcation), namely,

$$\rho_c = \frac{\sigma(\sigma+\beta+3)}{\sigma-\beta-1} \quad ,$$

which roughly equals 24.74 here. When $\rho$ is less than the critical value $\rho_c$ but greater than 1, then there are two stable nonzero stationary solutions. Figure 3.4 and Figure 3.5 illustrate trajectories at $\rho = 5$ and $\rho = 20$ respectively; we can see the orbit approach one of these equilibria.

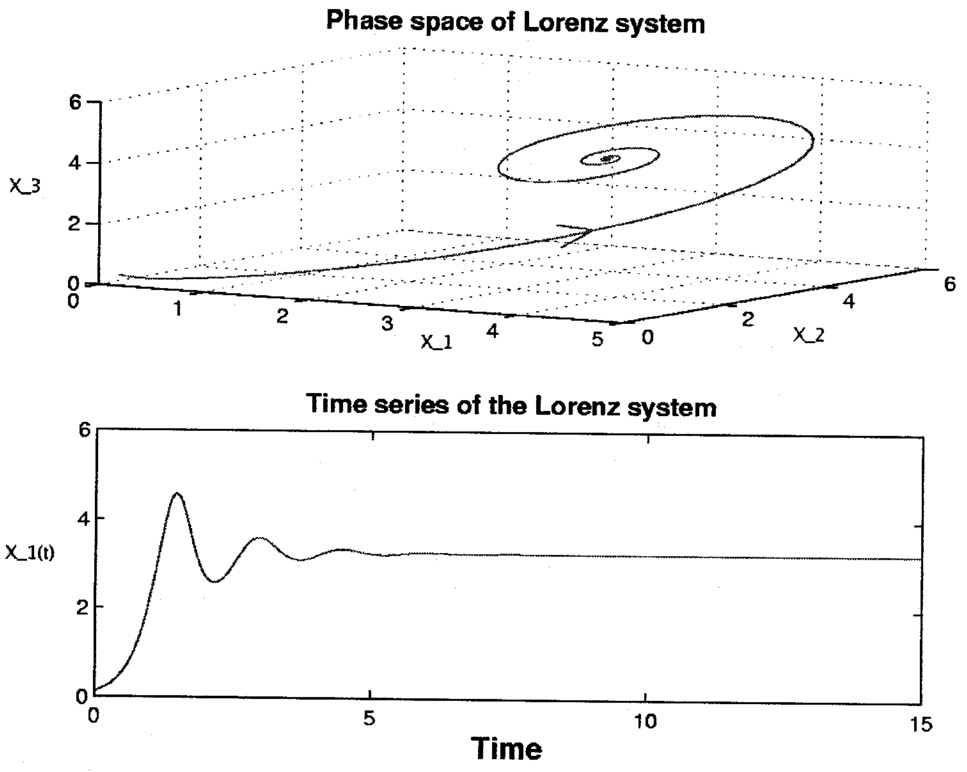


Figure 3.4: The Lorenz model: $\rho = 5$, step size= 0.03, # of steps= 500

As $\rho$ gets bigger, the solutions are more likely to approach a strange attractor than converge to a stationary solution. There is no apparent pattern to a "chaotic" solution, as it travels back and forth between the two wings of the attractor. This implies that it is difficult to predict the solutions at any future time based on the parameter values and the initial conditions. See Figure 3.6.

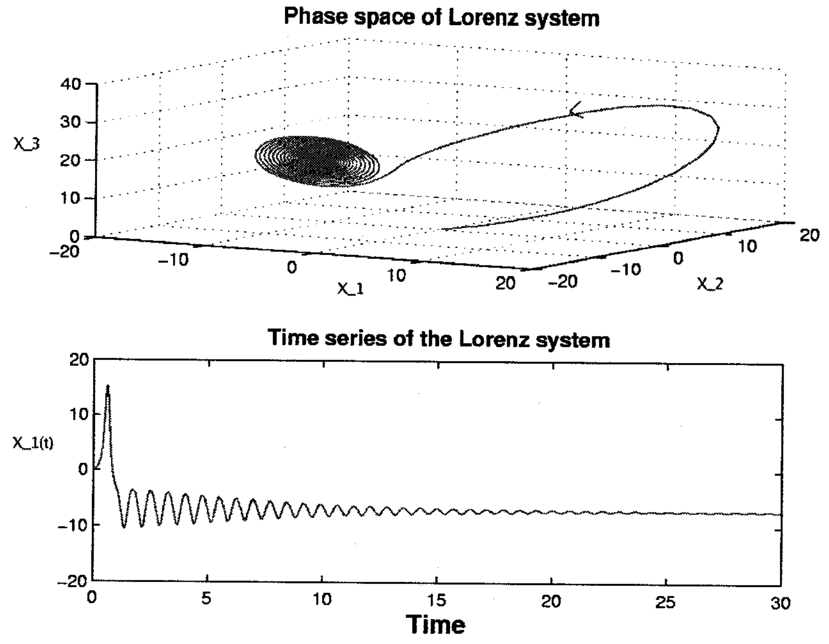


Figure 3.5: The Lorenz model: $\rho = 20$, step size= 0.02, # of steps= 1500

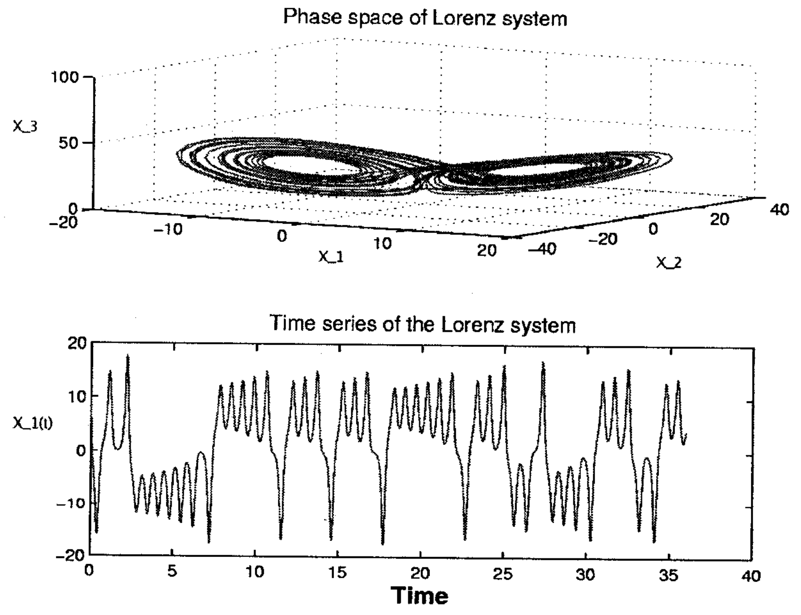


Figure 3.6: The Lorenz model: $\rho = 28$, step size= 0.02, # of steps= 1800

Figure 3.7 is the summary "bifurcation diagram" of the Lorenz system, computed with AUTO and plotted using PLAUT04 [82]. The $X$-axis represents the $\rho$ value, while the $Y$-axis represents the $x_1$ value. The color indicates the stability: blue is stable and red is unstable.

When $\rho < 1$, the zero solution is stable. At $\rho = 1$, one of the eigenvalues becomes positive and two new steady state solutions bifurcate. These are stable when $\rho < 24.74$. At $\rho = 24.74$, a pair of complex eigenvalues crosses the imaginary axis of the complex plane (a Hopf bifurcation), and the solutions become unstable and periodic solutions appear. Actually, the periodic orbits exist in the interval $\rho_r < \rho < \rho_c$, where $\rho_r \approx 13.9162$ . The value $\rho_r$ is often referred as a homoclinic explosion point; see [73]. Note that there is a second symmetry-related periodic solution branch in the lower half plane. However, we do not consider periodic solutions of the Lorenz system in this thesis, for more detail, one can refer to [28].

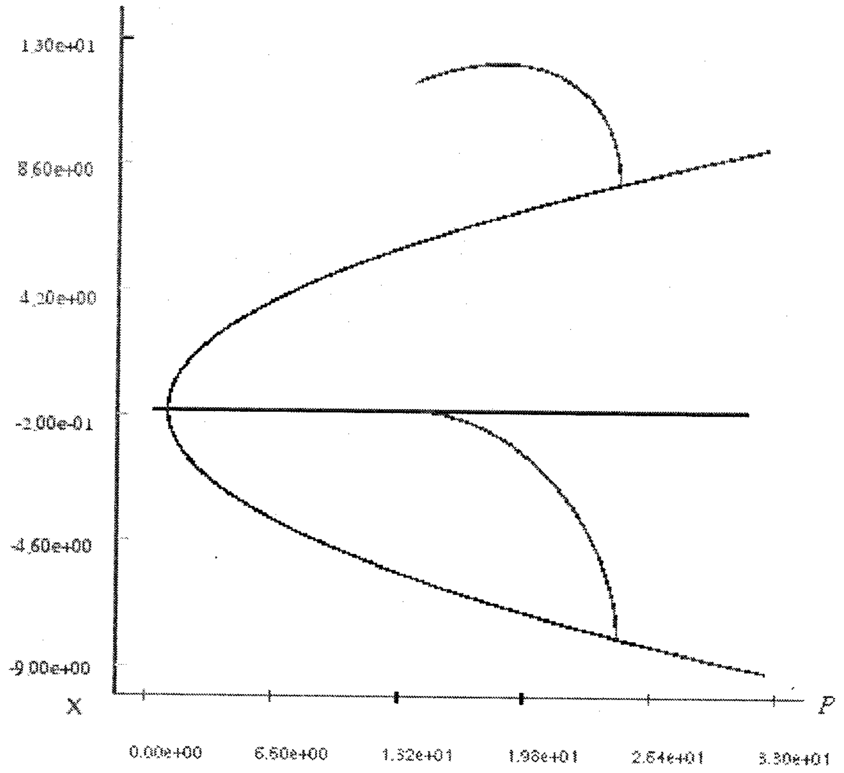


Figure 3.7: Bifurcation diagram of the Lorenz system

# Chapter 4

# Collocation Methods for Boundary Value Problems

## 4.1 Boundary value problems formulation

As discussed in Section 3.1, a differential equation can have initial conditions or boundary conditions. If conditions are specified at more than one value of the independent variable, they are referred to as *boundary conditions* and the combination of the differential equation and the boundary conditions is called a *boundary value problem*, *BVP* in short in this case.

Boundary value problems in ODEs arise in many applications, for example, they can be of the form

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t)) \quad , \quad t \in [0,1] \quad , \qquad (4.1)$$
$$\mathbf{x}(\cdot), \quad \mathbf{f}(\cdot,\cdot) \in \mathbf{R^n} \quad ,$$

with boundary conditions

$$\mathbf{b}_i(\mathbf{x}(0),\mathbf{x}(1)) = 0 \quad , \quad i = 1,2,\cdots,n \quad .$$

Sometimes integral constraints are also needed for specific applications; an example will appear in the following chapter.

Some parameter-dependent BVP examples are:

a) The Gelfand-Bratu Equation: [24].

$$x''(t) + \lambda e^{x(t)} = 0 \quad , \quad t \in [0,1] \quad , \tag{4.2}$$
$$x(0) = 0 \quad , \quad x(1) = 0 \quad .$$

which describes a *family* of boundary value problems, parametrized by a real parameter $\lambda$.

Figure 4.1 is the bifurcation diagram of the Gelfand-Bratu equation. Note that there are two solutions for $0 < \lambda < \lambda_c$, where $\lambda_c \approx 3.51$; there is one solution for $\lambda = \lambda_c$, and no solution for $\lambda > \lambda_c$.

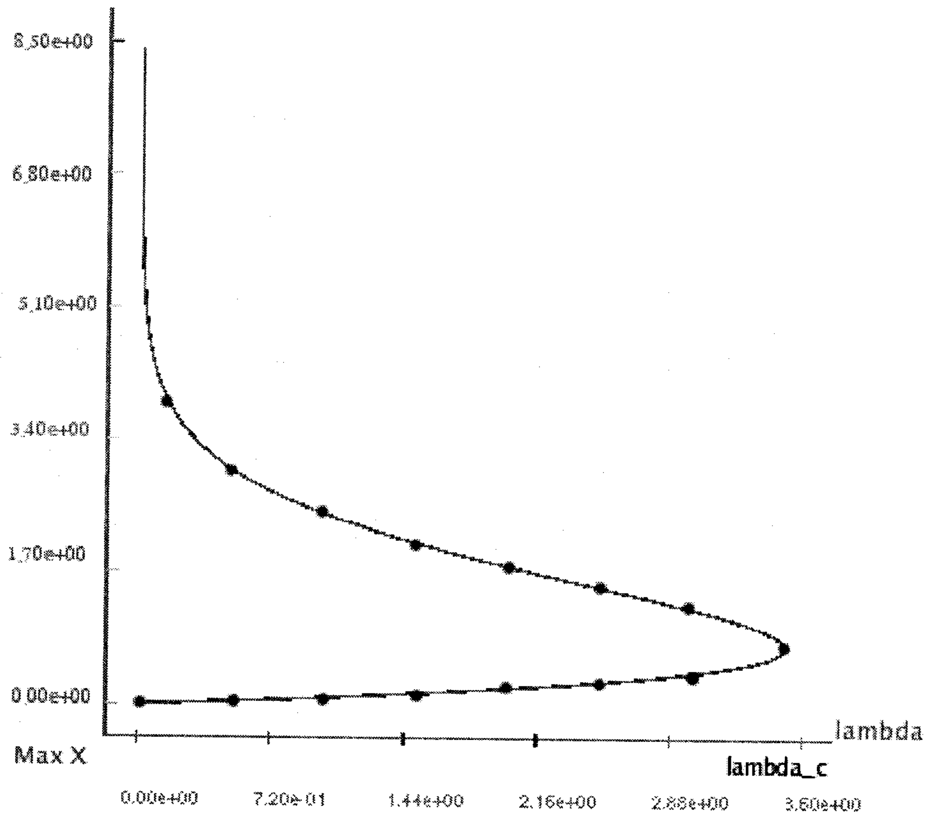


Figure 4.1: Bifurcation diagram of the Gelfand-Bratu equation

Figure 4.2 shows the corresponding solution diagram to the Gelfand-Bratu equation.

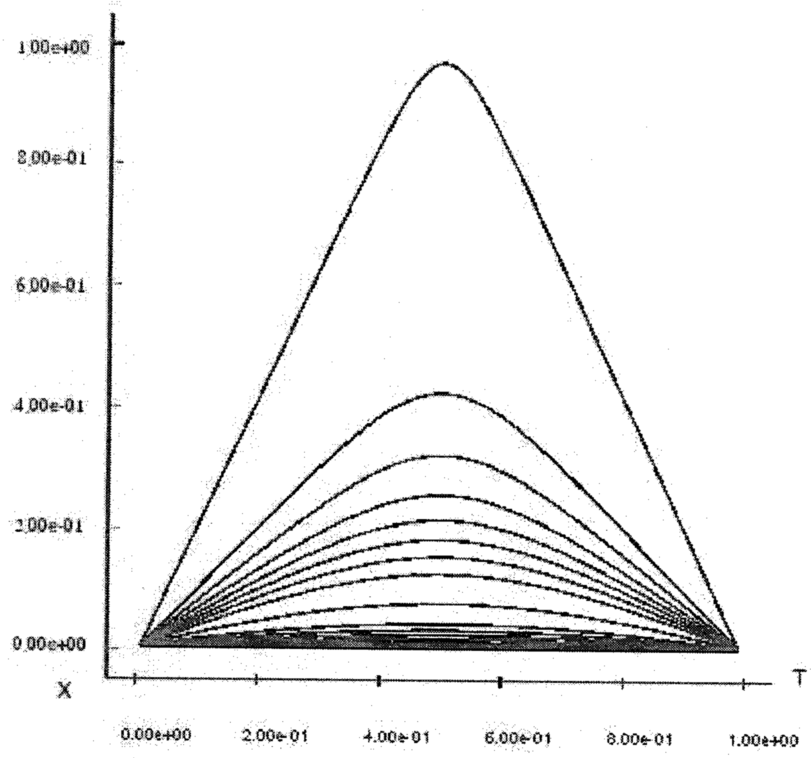


Figure 4.2: Some solutions to the Gelfand-Bratu equation

b) A nonlinear ODE eigenvalue problem: Consider the equation

$$x'' + (\lambda\pi)^2\, x - x^2 = 0 \,,$$

with boundary conditions $x(0) = x(1) = 0$, or when rewritten as a first order system

$$\begin{aligned} x_1' &= x_2 \;, \\ x_2' &= -(\lambda\pi)^2 x_1 + x_1^2 \;, \\ x_1(0) &= 0 \;,\quad x_1(1) = 0 \;. \end{aligned} \tag{4.3}$$

This equation also describes a *family* of boundary value problems, parametrized by a real parameter $\lambda$. A bifurcation diagram for this equation is shown in Figure 4.3).

When $\lambda$ is fixed at some particular value, say, $\lambda = 2.5$, then in addition the trivial solution, there are two more solutions located on other solution branches, namely, the solution labeled 3 on the first bifurcating branch and the solutions labeled 2 ( which actually represents two symmetry-related solutions ) on the second bifurcating branch.

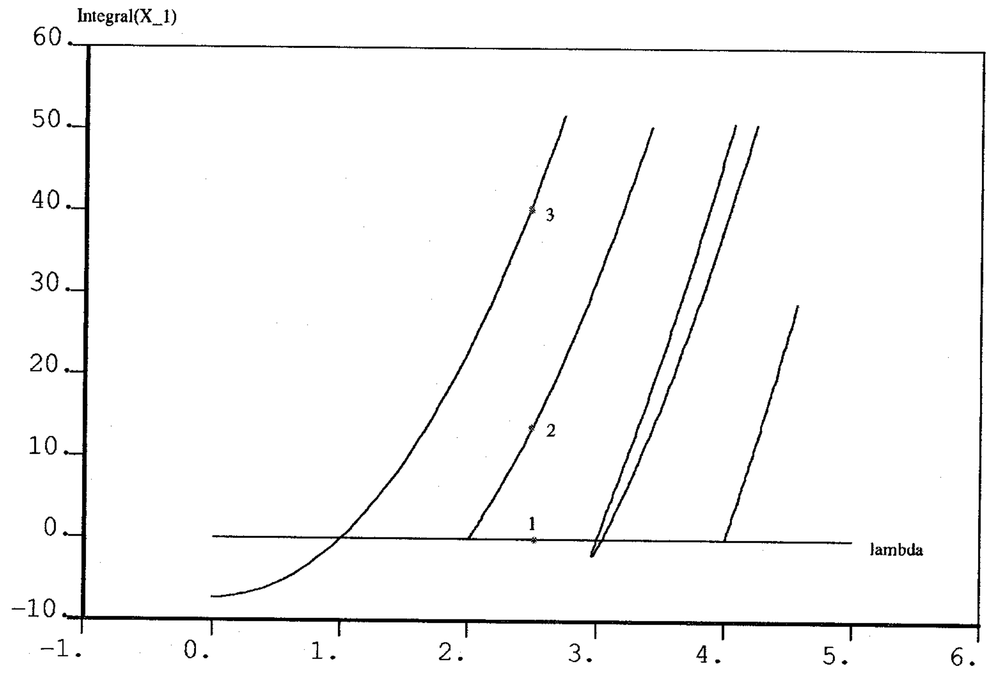


Figure 4.3: Bifurcation diagram of a nonlinear ODE eigenvalue problem

In this thesis, we also incorporate boundary conditions into the Lorenz equations, so that we can use the software AUTO for computing its stable and unstable manifolds.

Different methods can be used to solve BVPs, for example the shooting method, the finite difference method or the finite element method. See [65] for more details. In particular, AUTO uses orthogonal collocation with piecewise polynomials [22], which is a form of finite element method.

## 4.2 Collocation methods for boundary value problems

To solve boundary value problems (BVPs) of ordinary differential equations (ODEs), people often used finite difference methods. But finite difference methods typically cannot provide highly accurate solutions. At present, piecewise polynomial collocation is widely used for solving such problems. For example, it is the basic discretization in the software packages COLSYS [6], AUTO, *etc.* It determines a piecewise polynomial function that satisfies the differential equation at certain points. These points are known as *collocation points.* This method has several good characteristics:

- It can give very accurate solutions, especially when using good mesh adaption strategies.
- It is very efficient, compared to other methods.
- It is relatively easy to implement, and gives interpolation information, which is useful, for example, for graphical display.

As a simple example consider the equation

$$\begin{aligned} \mathbf{x}'(t) &= \mathbf{f}(\mathbf{x}(t)) \ , \quad t \in [0,1] \quad , \\ \mathbf{x}(0) &= 0 \quad , \end{aligned} \tag{4.4}$$

which is actually an IVP. Introduce a mesh

$$0 = t_0 < t_1 < \ldots < t_N = 1 \quad ,$$

with $t_j - t_{j-1} = \Delta t_j$. Further suppose there is only one collocation point $z_j$ in each interval $[t_{j-1}, t_j]$. In each interval $[t_{j-1}, t_j]$, the collocation method consists of determining a polynomial $\mathbf{p}_j$ of degree less than or equal to one, such that

$$\mathbf{p}_j(t_{j-1}) = \mathbf{x}_{j-1} \quad ,$$

and

$$\mathbf{p}_j'(z_j) = \mathbf{f}(\mathbf{p}_j(z_j)) \quad ,$$

for $z_j = \frac{1}{2}(t_{j-1} + t_j)$, *i.e.*, $z_j$ is the midpoint of the $j$th interval. Then set

$$\mathbf{x}_j = \mathbf{p}_j(t_j) \quad .$$

See Figure 4.4 for graphical illustration.

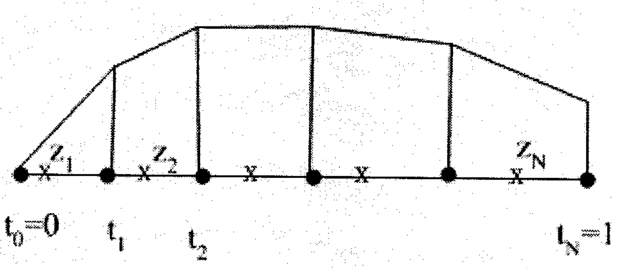


Figure 4.4: Collocation method for a simple IVP problem

This method is equivalent to the finite difference method

$$\begin{aligned} \frac{\mathbf{x}_j - \mathbf{x}_{j-1}}{\Delta t_j} &= \mathbf{f}(\frac{\mathbf{x}_{j-1} + \mathbf{x}_j}{2}) \ , \quad j = 1, 2, \cdots, N \quad , \\ \mathbf{x}_0 &= 0 \quad , \end{aligned} \tag{4.5}$$

known as the *Midpoint Rule.*

## 4.3 The use of collocation method in AUTO

Here we discuss the method of "*orthogonal collocation with piecewise polynomials* " used in AUTO, for solving boundary value problems. This method is very accurate and allows adaptive mesh-selection.

### 4.3.1 Orthogonal collocation

For the collocation method, the location of the collocation points in each subinterval $[t_{j-1}, t_j]$ has an important impact on how well the method works. Choosing evenly spaced points may be the first thought, but this does not yield the highest possible order of accuracy. Instead, choosing the collocation points as the zeros of a member of the family of orthogonal polynomials gives optimal order of accuracy. These points are called *Gauss points*, and the method is then referred to as *orthogonal collocation.*

Consider a first order system of ODEs

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t),\ \mu,\ \lambda)\ , \quad t \in [0,1] \quad , \tag{4.6}$$

where $\mathbf{x}(\cdot),\ \mathbf{f}(\cdot) \in \mathbf{R^n},\ \mu \in \mathbf{R^{n_\mu}},\ \lambda \in \mathbf{R}$. We can think of $\lambda$ as the "free" parameter, while the vector $\mu$ is part of the "solution" $(\mathbf{x}(\cdot), \mu)$. Assume the boundary conditions are of the form

$$\mathbf{b}(\mathbf{x}(0), \mathbf{x}(1), \mu, \lambda) = \mathbf{0}\ , \quad \mathbf{b}(\cdot) \in \mathbf{R^{n_b}}\ ,$$

and that there are also integral constraints of the form

$$\int_0^1 \mathbf{q}(\mathbf{x}(s), \mu, \lambda) ds = \mathbf{0}\ , \quad \mathbf{q}(\cdot) \in \mathbf{R^{n_q}}\ .$$

(For the case that $t \in [a, b]$, we can transform the time range to the interval $[0, 1]$). We require that

$$n_\mu = n_b + n_q - n \geq 0\ ,$$

where $n_b$ denotes the number of boundary conditions, while $n_q$ is the number of integral constraints. As before, we divide the time domain into a mesh

$$0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = 1\ ,$$

with

$$\Delta t_j = t_j - t_{j-1}\ , \quad j = 1, 2, \cdots, N\ .$$

Here, the time steps $\Delta t_j$, $j = 1, 2, \cdots, N$ need not to be equal.

Let $P^m$ be the function space of vector polynomials of degree less than or equal to $m$. Then we define the function space of piecewise vector polynomials $P_h^m$ as

$$P_h^m = \mathbf{p}_h \in C[0,1] \ : \ \mathbf{p}_h|_{[t_{j-1},t_j]} \in P^m \ .$$

The word "piecewise" means that each mesh interval has its own local vector polynomial.

For the collocation method, we want to find a piecewise vector polynomial $\mathbf{p}_h \in P_h^m$ , and a vector $\mu \in R^{n_\mu}$ , that satisfy the collocation equations

$$\mathbf{p}_h'(z_{j,i}) = \mathbf{f}(\mathbf{p}_h(z_{j,i} \ , \ \mu \ , \ \lambda)) \ , \quad j = 1, 2, \cdots, N, \quad i = 1, 2, \cdots, m \quad . \tag{4.7}$$

Furthermore, $\mathbf{p}_h$ must satisfy the boundary and integral conditions. See Figure 4.5 for the graphical representation.

Since each local polynomial is determined by $(m+1)n$ coefficients, the total number of degrees of freedom is $(m+1)nN + n_\mu$, assuming that $\lambda$ is fixed.

The total number of the equations is:

$$\begin{aligned} \text{collocation equations} \ &: \ mnN \quad , \\ \text{continuity equations} \ &: \ (N-1)n \quad , \\ \text{constraint equations} \ &: \ n_b + n_q \quad . \end{aligned}$$

Here, the number of the continuity equations is $(N-1)n$ because neighboring vector polynomials must have equal value at the interior mesh points. The number of equations is then equal to the number of unknowns, namely,

$$nmN + (N-1)n + n_b + n_q = (m+1)nN + n_b + n_q - n = (m+1)nN + n_\mu \quad .$$

If the exact solution $\mathbf{x}(t)$ is smooth enough, and using Gauss points as the collocation points, the global accuracy of the method is of order $m$, *i.e.*,

$$||\mathbf{p}_h - \mathbf{x}||_\infty = O(h^m) \quad .$$

However, at each mesh point $t_j$, the accuracy is much higher [7], namely,

$$max_j|\mathbf{p}_h(t_j) - \mathbf{x}(t_j)| = O(h^{2m}) \quad .$$

The vector $\mu$ is equally accurate. For more details, see [25].

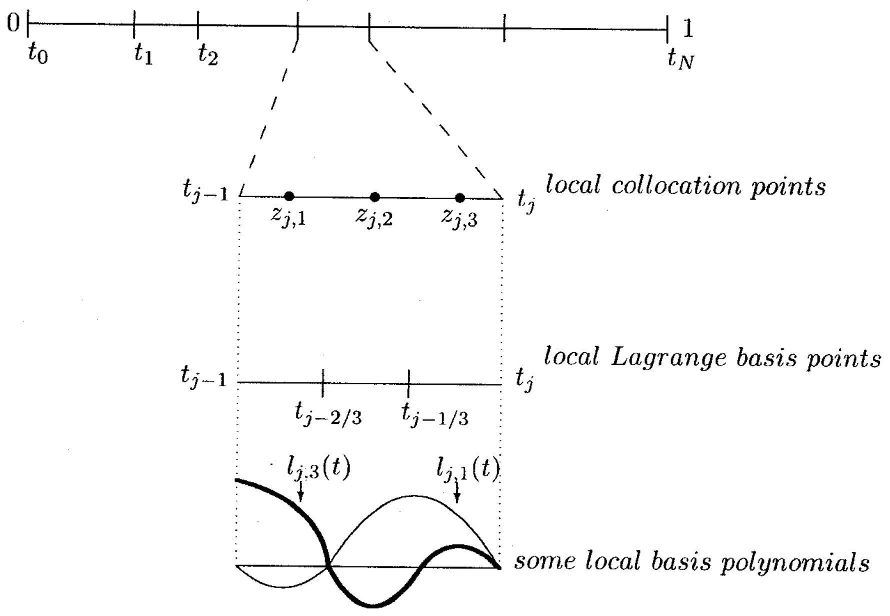


Figure 4.5: The mesh $\{ 0 = t_0 < t_1 < \cdots < t_N = 1 \}$. Collocation points are shown for the case $m = 3$.

## 4.4 Implementation of the orthogonal collocation method

Orbits are discretised in AUTO using the method of orthogonal collocation with piecewise polynomials, with 2-7 collocation points per mesh interval, see [5, 6, 13]. The mesh automatically adapts to the solution to equidistribute the local discretization error, [4, 9, 10, 68].

In AUTO, the collocation points in each mesh interval are Gauss points in order to have "superconvergence".

More specifically, for each subinterval $[t_{j-1}, t_j]$, we use the local Lagrange basis polynomials

$$\{l_{j,i}(t)\} \ , \quad j = 1, 2, \cdots, N \ , \ i = 0, 1, 2, \cdots, m \ ,$$

defined by

$$l_{j,i}(t) = \prod_{k=0, k\neq i}^{m} \frac{t - t_{j-\frac{k}{m}}}{t_{j-\frac{i}{m}} - t_{j-\frac{k}{m}}} \ , \tag{4.8}$$

where

$$t_{j-\frac{i}{m}} \equiv t_j - \frac{i}{m}\Delta t_j \ , \tag{4.9}$$

to represent the corresponding local polynomial

$$\mathbf{p}_j(t) = \sum_{i=0}^{m} l_{j,i}(t)\ \mathbf{x}_{j-\frac{1}{m}} \ . \tag{4.10}$$

See Figure 4.5 for an illustration of some Lagrange basis polynomials. With this choice of basis

$$\mathbf{x}_j \quad \text{will approximate} \quad \mathbf{x}(t_j) \quad \text{and} \quad \mathbf{x}_{j-\frac{i}{m}} \quad \text{will approximate} \quad \mathbf{x}(t_{j-\frac{i}{m}}) \ ,$$

where $\mathbf{x}(t)$ is the solution of the continuous problem.
The collocation equations are

$$\mathbf{p}_j'(z_{j,i}) = \mathbf{f}(\mathbf{p}_j(z_{j,i} \ , \ \mu \ , \ \lambda), \quad i = 1, 2, \cdots, m \ , \quad j = 1, 2, \cdots, N \ . \tag{4.11}$$

The discrete boundary conditions are

$$\mathbf{b}_i(\mathbf{x}_0, \mathbf{x}_N, \mu, \lambda) = 0 \ , \quad i = 1, \cdots, n_b \ . \tag{4.12}$$

The integrals can be discretized as

$$\sum_{j=1}^{N}\sum_{i=0}^{m}\omega_{j,i}\, q_k(\mathbf{x}_{j-\frac{i}{m}}\ ,\ \mu\ ,\ \lambda) = 0\ , \quad k = 1,\cdots,n_q \quad . \tag{4.13}$$

where the $\omega_{j,i}$ are the Lagrange quadrature coefficients.

# Chapter 5

# Numerical Continuation

## 5.1 Continuation of solutions

### 5.1.1 Introduction

Numerical continuation methods have been an important tool in the numerical solution of nonlinear systems. The methods may be used not only to compute solutions, but also to gain insight into qualitative properties of the solutions.

First we briefly discuss algorithms for computing families of solutions to nonlinear equations. The idea of continuation is as follows.

Consider a smooth function

$$\mathbf{F} : R^{n+1} \to R^n \ .$$

We want to compute a *solution family*, or *solution branch*, of the equation

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \ . \tag{5.1}$$

Numerical continuation is a technique to compute a sequence of points which approximate the desired solution family. The two main types of solutions that may be continued are:
(i) time-independent solutions (stationary states),
(ii) time-dependent solutions, subject to initial and/or boundary conditions.

While continuation of stationary states is rather straightforward, boundary value problems pose an additional difficulty as they need to be discretized. The branch of solutions obtained by continuation may contain bifurcation points, where two or more branches intersect, or points where the solution changes stability. By performing stability and bifurcation analysis during continuation, we can construct bifurcation diagrams. These diagrams are useful, since they give a detailed insight into various kinds of dynamics of the system studied.

There are several software packages for the numerical analysis of bifurcations in dynamical systems. AUTO is one of the earliest packages and perhaps the most widely used; other packages are briefly described in Appendix C; see also [3, 11, 24, 25, 51, 66] and [71].

Before we explore numerical continuation in more detail, we first discuss under what conditions a solution will actually persist, when problem parameters are changed.

### 5.1.2 The Implicit Function Theorem

The fundamental theoretical tool for numerical continuation is the *Implicit Function Theorem* (IFT), [41, 58], which we present here in a somewhat particular form [45, 46].

Let $\mathbf{F}$ be a smooth function

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad , \quad \mathbf{F} \, : R^{n+1} \to R^n \ . \tag{5.2}$$

Let $\mathbf{F}_\mathbf{x}(\mathbf{x}_0)$ denote the Jacobian matrix of $\mathbf{F}(\mathbf{x})$ evaluated at a solution $\mathbf{x}_0$. Note that the elements of $\mathbf{F}_\mathbf{x}(\mathbf{x}_0)$ are the derivatives of the $n$ component functions of $\mathbf{F}$ with respect to the $n+1$ variables represented by $\mathbf{x}$. Thus the matrix $\mathbf{F}_\mathbf{x}(\mathbf{x}_0)$ has $n$ rows and $n+1$ columns. If

$$Rank(\mathbf{F}_\mathbf{x}(\mathbf{x}_0)) = n \quad ,$$

or equivalently,

$$dim\ \mathcal{N}(\mathbf{F}_\mathbf{x}(\mathbf{x}_0)) = 1 \quad ,$$

where $\mathcal{N}$ denotes the nullspace of $\mathbf{F}_\mathbf{x}(\mathbf{x}_0)$,then the Implicit Function Theorem guarantees that there exists a unique family $\mathbf{x}(s)$, where $s \in R$, and a $\delta > 0$, such that

$$\mathbf{x}(0) = \mathbf{x}_0 \ , \qquad \mathbf{F}(\mathbf{x}(s)) = \mathbf{0} \quad \text{for} \quad |s| < \delta \quad .$$

### 5.1.3 Numerical continuation

From the above discussion it is clear that the IFT plays an important role in the design of continuation methods.

Consider the Equation 5.2. If the assumptions of the IFT are satisfied, *i.e.,* if

$$Rank(\mathbf{F}_{\mathbf{x}}(\mathbf{x}_0)) \ = \ n \quad ,$$

for a given solution $\mathbf{x}_0$, then there exists locally a family of solutions $\mathbf{x}(s)$ to Equation 5.2.

An extended system for computing "the next solution", $\mathbf{x}_1$, is given by

$$\begin{aligned} a) \qquad \mathbf{F}(\mathbf{x}_1) \quad &= \quad \mathbf{0} \quad , \\ b) \quad (\mathbf{x}_1 - \mathbf{x}_0)^*\dot{\mathbf{x}}_0 \quad &= \quad \triangle s \quad , \end{aligned} \tag{5.3}$$

where the superscript * denotes transpose, and where $\triangle s$ is a scalar, which is the step size in the continuation procedure. $\triangle s$ is set by the user, and normally adapted along the branch depending on the convergence history of Newton's method. $\dot{\mathbf{x}}_0$ is a null vector of the Jacobian matrix $\mathbf{F}_{\mathbf{x}}(\mathbf{x}_0)$; it is also the unit tangent to the path of solutions at $\mathbf{x}_0$, and can be computed very efficiently [21]. Figure 5.1 shows a geometrical interpretation of this method, generally known as *Keller's pseudo-arclength method* [45].

This method can be shown to work near a regular solution $\mathbf{x}_0$, *i.e.,* if the null space of $\mathbf{F}_{\mathbf{x}}(\mathbf{x}_0)$ is one-dimensional, as can be shown to follow from the IFT. In fact, in this case the Jacobian of Equation 5.2, evaluated at $\mathbf{x}_0$, i.e., the $n+1$ by $n+1$ matrix

$$\begin{pmatrix} \mathbf{F}_{\mathbf{x}}(\mathbf{x}_0) \\ \dot{\mathbf{x}}_0^* \end{pmatrix} , \tag{5.4}$$

is easily seen to be nonsingular. The solution branch can be parametrized locally by $\triangle s$.

In addition, provided $\triangle s$ is sufficiently small, and given a sufficiently accurate initial approximation to $\mathbf{x}_1$ (*e.g.*, $\mathbf{x}_1^{(0)} = \mathbf{x}_0 + \triangle s\ \dot{\mathbf{x}}_0$), it can be shown that Newton's method for solving Equation 5.3 converges. Bifurcation points along the solution branch can be located accurately, as they correspond to singularity of the Jacobian matrix 5.4. There exist standard algorithms for switching branches at bifurcation points. These algorithms have been implemented in AUTO [24].

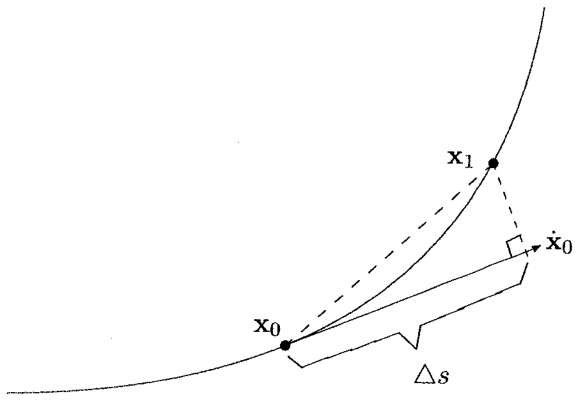


Figure 5.1: Graphical representation of the pseudo-arclength method

Chapter 6

# Computation of Stable/Unstable Manifolds in the Lorenz Equations

## 6.1 Overview

Stable and unstable manifolds associated with equilibrium points are important objects in phase portraits. At first thought, it may seem that the computation of such manifolds can be carried out easily by numerical integration. For one-dimensional manifolds this is often true, as we discussed in Section 1.4. However, higher dimensional manifolds are more difficult to compute.

The Lorenz system introduced in Section 2.6 exhibits characteristics of complex geometric objects. More precisely, for the standard system parameters, the origin has a two-dimensional stable manifold and the other two equilibria each have a two-dimensional unstable manifold. The intersections of these two manifolds in the three-dimensional phase space form *heteroclinic connections* (see Chapter 7 for details) from the non-zero equilibrium to the origin.

The two-dimensional stable manifold of the origin for the Lorenz system, with the "standard" parameter values, is known as the *Lorenz manifold*. There are two difficulties in computing this manifold [40]:

- The stable eigenvalues at the origin of this system are approximately $-2.667$ and

$-22.828$, with a ratio that is approximately 8.56. Trajectories in the manifold tend to follow the weakly stable direction, which makes it difficult to "cover" the full manifold, even relatively near the origin.

- Part of the global manifold spirals around the $z-$axis while other parts of it curl around the stable manifolds of the equilibria located at approximately $(\pm 8.485, \pm 8.485, 27)$.

The manifold approaches itself arbitrarily closely, in fact infinitely often, in certain areas of phase space. This makes it practically impossible to reliably compute a large portion of the manifold with techniques that advance its computational boundary based on local information near this boundary. Such methods are prone to "sheet jumping".

Many papers address the geometry of the Lorenz manifold, because it has a number of astonishing properties. Perelló [62] computed the stable manifold of the origin, and also gave a sketch for $\rho$ close to the critical value $\rho_c$. He also considered the unstable manifold of the non-zero equilibria. It was obtained by following a line segment, which is quite close to our approach of computing the unstable manifold of the non-zero equilibria.

Thompson and Stewart [75] computed trajectories that illustrate the local stable manifold. Based on this, a more advanced visualization was done by Stewart [74]; the dynamics and global bifurcations of the Lorenz system can be observed there in the three-dimensional phase space.

The first hand-drawn image of the Lorenz manifold computed under the standard parameter values appeared in the book of Abraham and Shaw [1] in 1985, while Guckenheimer and Worfolk's article [39] is the first paper that has computer-generated images of the Lorenz manifold. For more detail, see [50].

In this chapter, AUTO is used to compute the above-mentioned stable and unstable manifolds of the Lorenz system. Corresponding diagrams are presented either by using the software package PLAUT04 [28, 82] or by using the new visualization tool VMD.

Below, in Figure 6.1, is the bifurcation diagram of the Lorenz equations, reproduced here to review some properties of the Lorenz system.

The figure shows that the zero solution is unstable for $\rho > 1$. The two nonzero stationary solutions bifurcate at $\rho = 1$ and become unstable for $\rho > \rho_c \approx 24.74$. At $\rho_c$ there are Hopf bifurcations and unstable periodic solutions emanate from each Hopf bifurcation. When $\rho > \rho_c$, the famous Lorenz attractor appears.

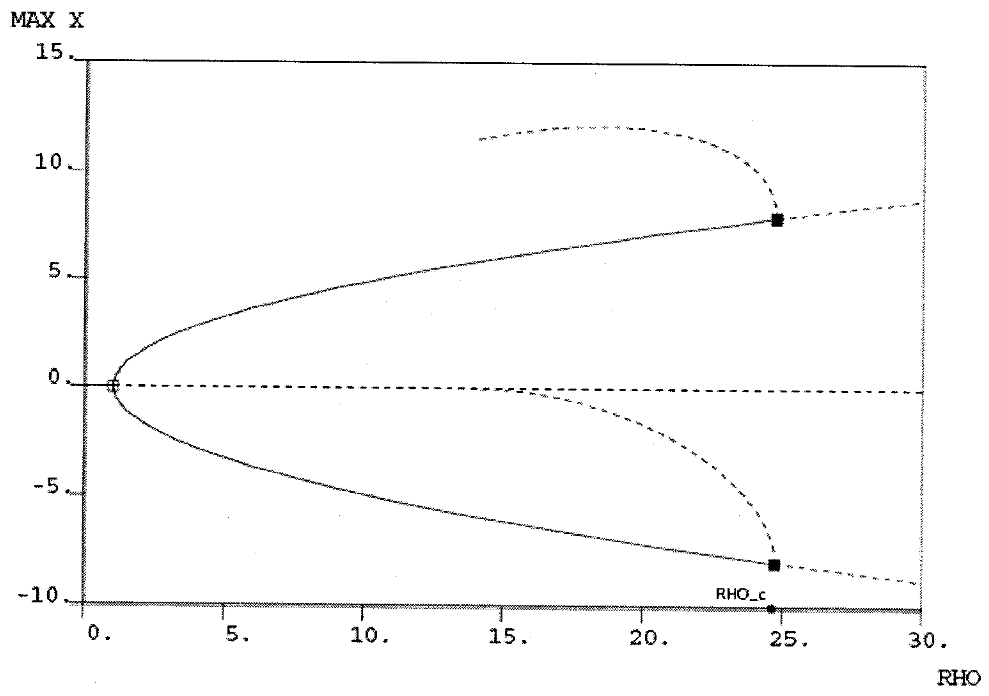


Figure 6.1: Bifurcation diagram of the Lorenz equations: solid curves denote stable solutions and dashed curves denote unstable solutions.

The procedure for computing stable or unstable manifolds in this thesis is based upon the solution of "boundary value problems". The idea is to compute the family of trajectories that form the manifold by numerical continuation, in which each step consists of finding the solution of a differential equation subject to initial *and global* constraints [50]. The basic idea is presented in the following section.

## 6.2 The stable manifold of the origin

### 6.2.1 Computing the stable manifold of the origin

The method for computing the stable manifold of the origin uses numerical continuation. Consider the ODEs representing the Lorenz equations, using the standard parameter values, written as

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}(t)) \quad , \quad t \in [0,T] \quad . \tag{6.1}$$

Since the integration time $T$ will be variable, we introduce a new, scaled time variable $\hat{t}$ to let the integration always takes place over the interval $[0,1]$. The transformation is as follows

$$\hat{t} \equiv \frac{t}{T} , \quad \hat{t} \in [0,1] \quad .$$

Then

$$\mathbf{x}'(t) = \frac{d\mathbf{x}}{dt} = \frac{d\mathbf{x}}{d\hat{t}}\frac{d\hat{t}}{dt} = \frac{d\mathbf{x}}{d\hat{t}}\frac{1}{T} \quad .$$

Let

$$\hat{\mathbf{x}}(\hat{t}) = \mathbf{x}(t(\hat{t})) \quad ,$$

so that

$$\frac{d\hat{\mathbf{x}}}{d\hat{t}}\frac{1}{T} = f(\hat{\mathbf{x}}(\hat{t})) \quad ,$$

or

$$\frac{d\hat{\mathbf{x}}}{d\hat{t}} = T\, f(\hat{\mathbf{x}}(\hat{t})) \quad .$$

Dropping the "ˆ", the transformed equation is

$$\mathbf{x}'(t) = T\, \mathbf{f}(\mathbf{x}(t)) \, , \quad t \in [0,1] \quad . \tag{6.2}$$

Note that the original integration time $T$ is now an explicit variable in the equations.

The origin $\mathbf{0} = (0,0,0)^T$ is a saddle point with eigenvalues

$$\mu_1 \approx -2.66 \, , \quad \mu_2 \approx -22.8 \, , \quad \mu_3 \approx 11.82 \quad ,$$

and corresponding normalized eigenvectors

$$\mathbf{v}_1 \, , \quad \mathbf{v}_2 \, , \quad \mathbf{v}_3 \quad .$$

The computing procedure is as follows. First, let us suppose that an initial orbit $\mathbf{x}_0(t)$ has already been computed, for $t$ from 0 to $T_0$ (where $T_0 < 0$, since we deal with a *stable* manifold here), with

$$\mathbf{x}_0(0) \quad \text{close to the origin} \quad \mathbf{0} \quad ,$$

and

$$\mathbf{x}_0(0) \quad \text{in the stable eigenspace spanned by} \ \ \mathbf{v}_1 \ \text{and} \ \mathbf{v}_2 \quad ,$$

or, more precisely,

$$\mathbf{x}_0(0) \;=\; \mathbf{0} \;+\; \epsilon \left( \frac{\cos(\theta)}{|\mu_1|}\, \mathbf{v}_1 \;-\; \frac{\sin(\theta)}{|\mu_2|}\, \mathbf{v}_2 \right) \quad ,$$

where $\epsilon$ is a small radius in the stable eigenspace centered at the origin. Starting data include a value of $\theta$ between 0 and $2\pi$. Say, if we take $\theta = 0$, then the initial orbit satisfies Equation (6.2) and

$$\mathbf{x}_0(0) \;=\; \frac{\epsilon}{|\mu_1|}\, \mathbf{v}_1 \quad .$$

The starting orbit $\mathbf{x}(t)$ has length

$$L_0 \;=\; T_0 \int_0^1 \|\, \mathbf{f}(\mathbf{x}_0(s))\, \|\, ds \quad .$$

In other words, for given $\epsilon = \epsilon_0$ and $L = L_0$, the initial orbit is a solution

$$X_0 \;=\; (\, \mathbf{x}_0(\cdot)\, ,\; \theta_0\, ,\; T_0\, ) \quad , \quad \text{where} \quad \theta_0 \;=\; 0 \quad ,$$

of the equation

$$F(X) \;=\; 0 \quad ,$$

where

$$X \;=\; (\, \mathbf{x}(\cdot)\, , \quad \theta\, , \quad T\, ) \quad ,$$

and where $F(X)$ can be written as

$$F(X) \equiv \begin{cases} \mathbf{x}'(t) - T\,\mathbf{f}(\mathbf{x}(t)) \\ \mathbf{x}(0) - \epsilon\,(\frac{\cos(\theta)}{|\mu_1|})\,\mathbf{v}_1 - \frac{\sin(\theta)}{|\mu_2|}\,\mathbf{v}_2\,) \\ T\,\int_0^1 \|\,\mathbf{f}(\mathbf{x}(s))\,\|\,ds - L \end{cases} \tag{6.3}$$

Given $X_0$ and $\dot{X}_0$, with $\|\,\dot{X}_0\,\| = 1$, pseudo-arclength continuation, as discussed in Chapter 5, can now be used to compute a next solution $X_1$, by solving

$$F(X_1) = 0 \quad ,$$

$$(X_1 - X_0)^*\,\dot{X}_0 - \Delta s = 0 \quad ,$$

where

$$X_1 = (\,\mathbf{x}_1(\cdot)\,,\quad \theta_1\,,\quad T_1\,) \quad ,$$

keeping $L$ and $\epsilon$ fixed.

A key point in the computation is that the step size $\Delta s$ in the continuation procedure measures the change of the *entire trajectory* (also including the change of the parameters $\theta$ and $T$), and not just the change in the initial conditions [50]. This fact generally results in a reasonable distribution of trajectories along the stable manifold. Since the continuation procedure is stepwise, with a small change, $\Delta s$, of the solution $X$ in each continuation step, the entire manifold is covered; *i.e.*, "jumps" of the solutions generally cannot occur.

### 6.2.2 Variations on the computational method

There are possible variations on the basic numerical scheme. For example, instead of fixing $L$ and allowing $T$ to vary, one can fix $T$ and allow $L$ to vary. In this case, $F(X)$ still takes the form of Equation (6.3), but with $X = (\mathbf{x}(\cdot),\ \theta,\ L)$ for given $\epsilon$ and $T$.

Alternatively, one can fix one coordinate or a function of the coordinates at a particular value to constrain the end point, and again free both $T$ and $\theta$. This is done by including an appropriate functional $g$ [50]; thus $F(X)$ becomes

$$F(X) \equiv \begin{cases} \mathbf{x}'(t) \ - \ T\,\mathbf{f}(\mathbf{x}(t)) \\ \mathbf{x}(0) \ - \ \epsilon\,(\ \frac{\cos(\theta)}{|\mu_1|})\ \mathbf{v}_1 \ - \ \frac{\sin(\theta)}{|\mu_2|}\ \mathbf{v}_2\ ) \\ g(\mathbf{x}(1)\ ,\ T) \ - \ \alpha \end{cases} , \qquad (6.4)$$

where

$$X \ = \ (\ \mathbf{x}(\cdot)\ ,\ \theta\ ,\ T\ ) \quad , \quad (\ \text{for given } \alpha\ ) \quad .$$

In addition, it is also possible to use a combination of end-point conditions and integral constraints, as will be seen in later sections.

### 6.2.3 The starting procedure

In the previous sections, we assumed that an initial orbit had already been computed. In fact, the initial orbit can be computed quite easily by numerical integration. However, we use *continuation*, even for the computation of the initial orbit. The main reason for this approach is that the numerical results produced by AUTO will then be compatible as starting data for the principal step in the algorithm, which was described in the preceding section.

To compute an initial orbit, we take $F(X)$ as the system (6.3), where

$$X \ = \ (\ \mathbf{x}(\cdot)\ ,\ L\ ,\ T\ ) \quad , \quad (\ \text{for given } \epsilon \text{ and } \theta = 0\ ) \quad .$$

Starting data are $\mathbf{x}(t) \equiv \frac{\epsilon}{|\mu_1|}\ \mathbf{v}_1$, (*i.e.*, $\mathbf{x}(\mathrm{t})$ is constant), a very small value of $T$ ($T < 0$), and $L = 0$. Stopping this continuation at a specified value of $L$ then yields an initial orbit. See Figure 6.2 for an example of an initial orbit obtained by this method.

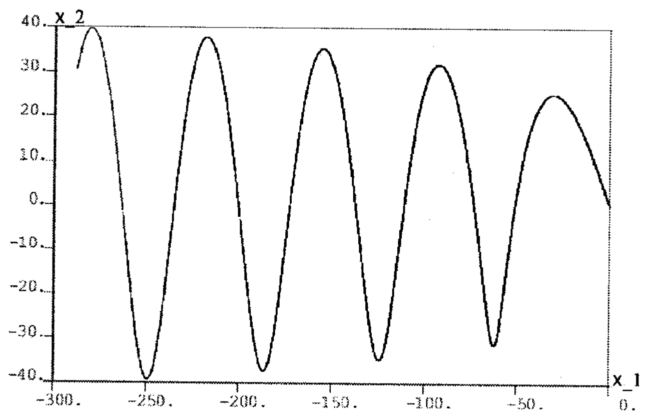


Figure 6.2: An initial orbit for computing the stable manifold of the origin of the Lorenz system; obtained by fixing $\epsilon = 5.0$, $\theta = 0$, and letting $T$ and $L$ vary, stopping at $L = 1000$.

### 6.2.4 Numerical results

As mentioned above, once an initial orbit of desired length is obtained, we then restart from this solution. We let the angle $\theta$ and the integration time $T$ change, while fixing $L$ and $\epsilon$. After a specified number of such continuation steps, we can see part of the manifold; see Figure 6.3 for an example.

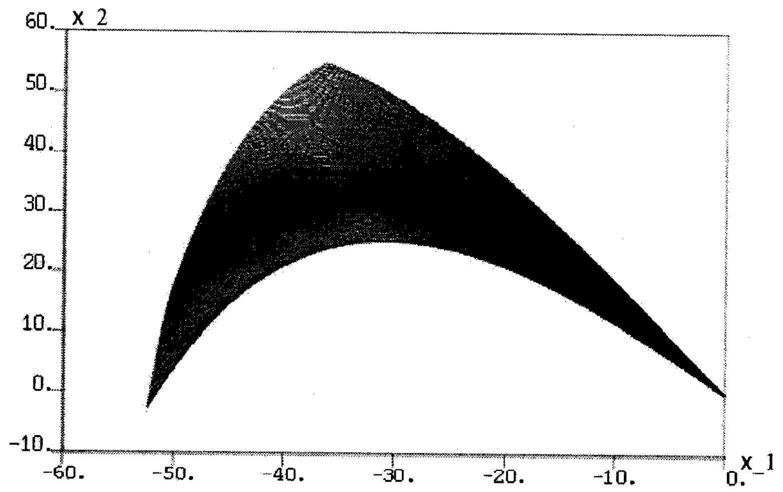


Figure 6.3: Part of the stable manifold of the origin computed by continuation; with $\epsilon = 5.0$ (which is "small" here), $\mu_1 \approx -2.667$ and $\mu_2 \approx -22.828$, obtained by first increasing $L$ and $T$, keeping $\theta = 0$, then fixing $L$ and freeing $\theta$, stopping after 100 continuation steps.

After continuing for a sufficiently large number of steps, part of the Lorenz manifold appears as illustrated in Figure 6.4.

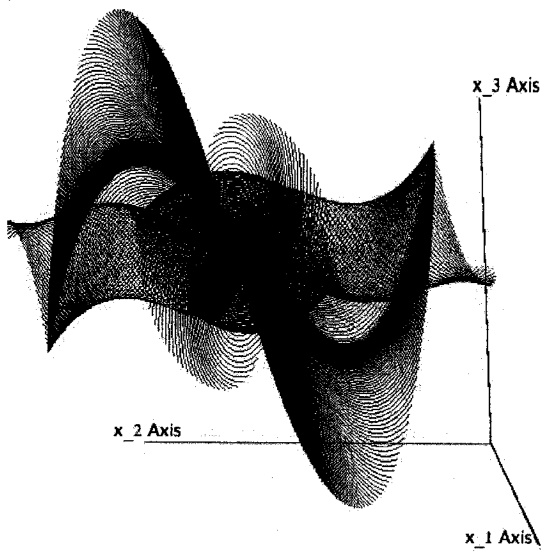


Figure 6.4: The Lorenz manifold obtained by first increasing $L$ and $T$, keeping $\theta = 0$, and then fixing $L$ and freeing $\theta$, stopping at $\theta = 2\pi$.

Figure 6.5 shows an enlargement near the origin of orbits of "short" length $L$ ($L = 100$), that were continued on the Lorenz manifold, the angle $\theta$ varies from 0 to $2\pi$, $L$ is fixed and $T$ is variable. Note that it appears that the length $L$ of the orbits is not fixed in the figure; however, this is due to the scaling and the view point.

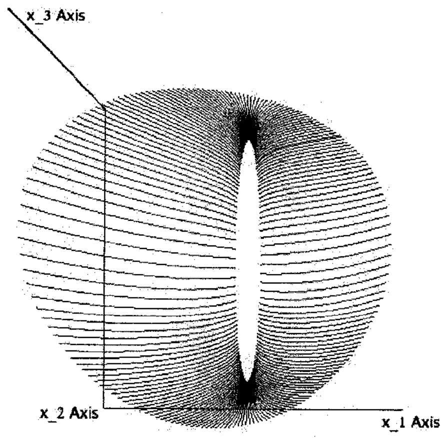


Figure 6.5: Short trajectories on the stable manifold near the origin, with $(\mathbf{x}(\cdot), T, \theta)$ variable and $L$ fixed.

As mentioned before, there are other ways to do the continuation. Figure 6.6 shows a part of interest of the stable manifold; the $x_1$ component is fixed at the end point of the trajectories. Figure 6.7 illustrates the same surface, showing the actual orbits. Note that all trajectories in Figure 6.7 emanate from near the origin.

Figure 6.7 illustrates the flexibility of the continuation method. Instead of fixing $T$, one can fix the value of a component of the end point $\mathbf{x}(1)$ and let $\theta$ vary. A scroll-like portion of the stable manifold is generated this way.

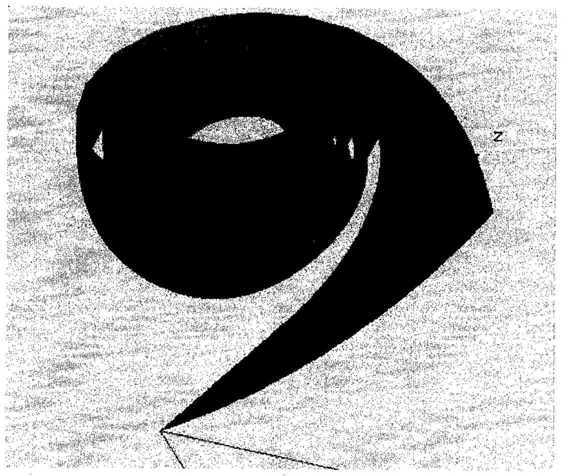


Figure 6.6: Part of the Lorenz manifold, here plotted as a surface; $x_1(1)$ is fixed

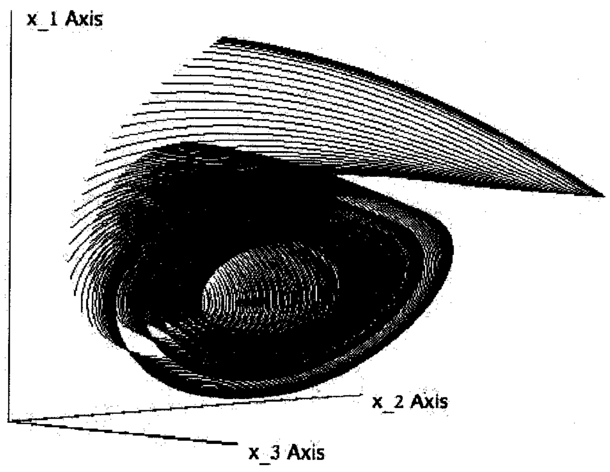


Figure 6.7: Part of the Lorenz manifold, here represented by orbits; $x_1(1)$ is fixed

## 6.3 The unstable manifold of the non-zero stationary points

### 6.3.1 Computing the unstable manifold of the non-zero stationary points

For the Lorenz system, the two non-zero equilibria, denoted by $\mathbf{x}_{sp}^-$ and $\mathbf{x}_{sp}^+$ respectively, are approximately at

$$(x_1,\ x_2,\ x_3)\ =\ (\pm 8.485\ ,\ \ \pm 8.485\ ,\ 27)\quad .$$

As discussed in Section 2.6.4, the Jacobian at each of these equilibria has a pair of complex conjugate eigenvalues with positive real part, which correspond to a two-dimensional unstable manifold, and one negative real eigenvalue, corresponding to a one-dimensional stable manifold. Suppose again the corresponding normalized eigenvectors are

$$\mathbf{v}_1\ ,\quad \mathbf{v}_2\ ,\quad \mathbf{v}_3\quad .$$

We take $\mathbf{x}(0)$ near $\mathbf{x}_{sp}$, one of the non-zero stationary points, and we require $\mathbf{x}(0)$ to be in the unstable eigenspace spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$.

The computational set-up is somewhat similar to that of the stable manifold of the origin. Families of orbits are computed as solutions of two-point boundary value problems with AUTO. However, in the current case, we introduce additional parameters in the continuation procedure to do the computation.

First suppose that $\mathbf{w}_0$ is the eigenvector associated with the negative real eigenvalue of the transpose Jacobian at $\mathbf{x}_{sp}$, *i.e.*, of $\mathbf{f}_{\mathbf{x}}(\mathbf{x}_{sp})^*$, which is orthogonal to the two-dimensional unstable eigenspace $E_{sp}^u$; and $\mathbf{w}_1$ is the eigenvector associated with the real positive eigenvalue of the transpose Jacobian at the origin, *i.e.*, of $\mathbf{f}_{\mathbf{x}}(\mathbf{0})^*$, which is orthogonal to the two-dimensional stable eigenspace $E_0^s$. Suppose the starting point is $\mathbf{x}(0)$, the distance from $\mathbf{x}(0)$ to the non-zero point $\mathbf{x}_{sp}$ is $d_0$. We define the value of the inner product $< \mathbf{x}(0) - \mathbf{x}_{sp}\ ,\ \mathbf{w}_0 >$ to be $\tau_0$; if $\tau_0 = 0$, then $\mathbf{x}(0)$ lies in $E_{sp}^u$.

Similarly, if the distance from the endpoint $\mathbf{x}(1)$ to the origin is $d_1$, then the inner product $< \mathbf{x}(1) - \mathbf{0}\ ,\ \mathbf{w}_1 >$, denoted by $\tau_1$, measures whether the endpoint $\mathbf{x}(1)$ passes through the stable eigenspace of the origin. If $\tau_1 = 0$, then $\mathbf{x}(1)$ lies in $E_0^s$.

In addition, the derivative of the third solution component, $x_3'(0)$ at the starting point, we call it $\psi_0$, plays a role in the computation of an initial orbit. The starting procedure is designed to set the value of $\psi_0$ to zero, to ensure that the starting vector always lies in the unstable eigenspace $E_{sp}^u$ associated with $\mathbf{x}_{sp}$ and stays along a given curve in this eigenspace.

See Figure 6.8 for the graphical representation.

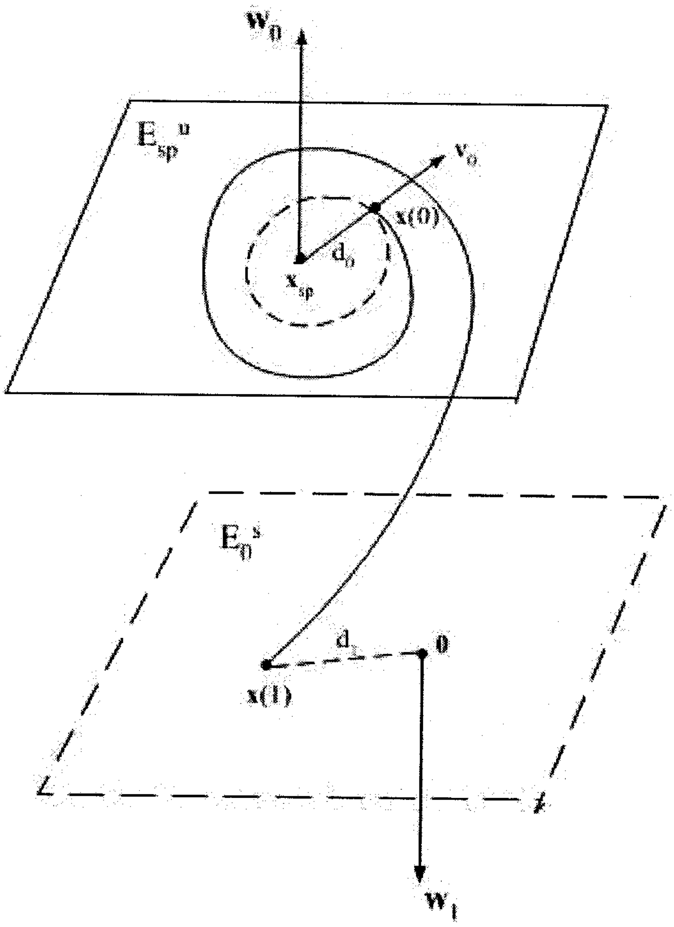


Figure 6.8: Graphical representation of the computation of the unstable Lorenz manifold

For the unstable manifold of the Lorenz system, $F(X)$ is defined by

$$F(X) \equiv \begin{cases} \mathbf{x}'(t) - T\,\mathbf{f}(\,\mathbf{x}(t)\,,\,\rho\,) \\ <\mathbf{x}(0) - \mathbf{x}_{sp}\,,\,\mathbf{w}_0> \; - \tau_0 \\ <\mathbf{x}(1) - \mathbf{0}\,,\,\mathbf{w}_1> \; - \tau_1 \\ ||\,\mathbf{x}(0) - \mathbf{x}_{sp}\,|| - d_0 \\ ||\,\mathbf{x}(1) - \mathbf{0}\,|| - d_1 \\ T\int_0^1 ||\,\mathbf{f}(\mathbf{x}(s))\,||\,ds \; - L \\ f_3\,(\,\mathbf{x}(0)\,) - \psi_0 \end{cases} \tag{6.5}$$

In this case there are 6 constraints, and consequently the number of required free parameters $n_p$ is

$$n_p = (\,n_b + n_q\,) - n_d + 1 = (\,5 + 1\,) - 3 + 1 = 4 \quad ,$$

where $n_b$ is the number of boundary conditions, $n_q$ is the number of integral constraints, and $n_d$ is the dimension of the system. Generically, the number of free parameters should be one more than the total constraints minus the dimension of the system.

### 6.3.2 The starting procedure

The starting procedure is somewhat more complicated than that for the stable manifold of the origin. First, we set initial approximate solutions for small $T_0$. For example, we take $T_0 = 0.00005$ and $d_0 = 0.5$.

We initialize $\mathbf{x}(t)$ to be constant, namely,

$$\mathbf{x}(t) \equiv \mathbf{x}_{sp} + d_0\,\mathbf{v}_0 \quad ,$$

where $\mathbf{v}_0$ is any vector in $E^u_{sp}$, with $||\mathbf{v}_0|| = 1$. Then $||\mathbf{x}(0) - \mathbf{x}_{sp}|| = d_0$, and we initialize $d_1 = ||\mathbf{x}(1) - \mathbf{0}|| = ||\mathbf{x}_{sp} + d_0\mathbf{v}_0||$, $\tau_1 =< \mathbf{x}(1) - \mathbf{0}, \mathbf{w}_1 >$, $L = 0$, and $\psi_0 = f_3(\mathbf{x}(0))$.

The eigenvector $\mathbf{w}_1$ associated with the real positive eigenvalue of the transpose Jacobian at the origin, is given by

$$\mathbf{w}_1 = \begin{pmatrix} -1 \\ (1 - \sigma - \sqrt{(\sigma - 1)^2 + 4\rho\sigma})/(2\rho) \\ 0 \end{pmatrix} .$$

The list of continuation parameters and the actual detailed implementation in AUTO can be found in Appendix A.

### 6.3.3 The continuation procedure and numerical results

There are several runs in the continuation process. Each run is addressed in detail below.

**Run** 1:

Taking $F(X)$ as the system (6.5), with

$$X = (\mathbf{x}(\cdot), T, L, \tau_1, d_1) , \quad (\text{ for fixed } \rho,\ \tau_0,\ d_0 \text{ and } \psi_0) .$$

In this case, $\tau_0 = 0$, $d_0$ is small, and $\psi_0 = f_3(\mathbf{x}(0))$, which is also small but not 0. In this run $\mathbf{x}(0)$ is in $E^u_{sp}$, and stays at a fixed location on the small circle of radius $d_0$ around $\mathbf{x}_{sp}$. Using continuation we compute an initial orbit until a specified value of $L$. See Figure 6.9 for a schematic representation.

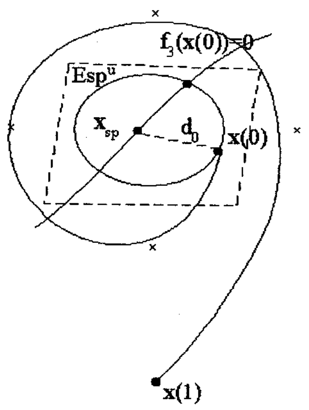


Figure 6.9: Illustration of the starting procedure: $f_3(\mathbf{x}(0)) = \psi_0$, which is small, but not equal to 0.

**Run** 2:

In this run we adjust the initial condition by letting the derivative of the third solution component $x_3'(0)$ vary, also freeing $L$ , $T$ and $\tau_1$, and keeping $\rho$, $\tau_0$, $d_0$ and $d_1$ fixed. So now $X$ takes the form

$$X \ = \ ( \ \mathbf{x}(\cdot) \ , \ T \ , \ L \ , \ \psi_0 \ , \ \tau_1 \ ) \quad , \quad ( \text{ for fixed } \rho, \ \tau_0, \ d_0 \text{ and } d_1 \ ) \quad .$$

We continue this family of orbits until $\psi_0 = 0$, *i.e.*, the starting point $\mathbf{x}(0)$ moves around the circle of radius $d_0$ until it reaches one of the two points where $f_3(\mathbf{x}(0))$ is zero. See Figure 6.10 for an illustration.

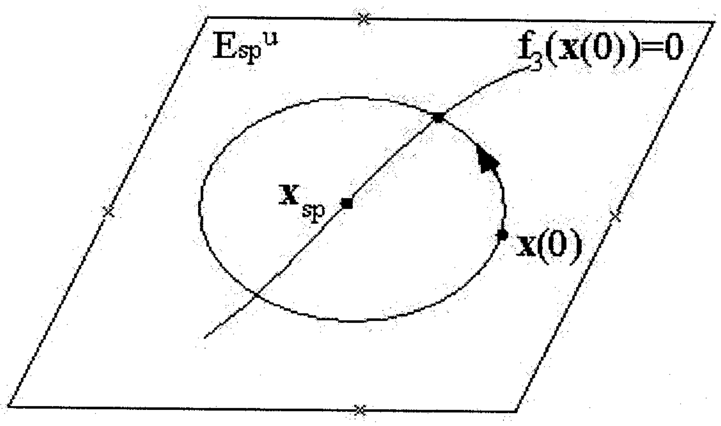


Figure 6.10: Illustration of adjusting the starting vector to lie along a certain curve

**Run** 3:

We repeat the procedure in Run 1, so $X$ still takes the form

$$X \ = \ ( \ \mathbf{x}(\cdot) \ , \ T \ , \ L \ , \ \tau_1 \ , \ d_1 \ ) \quad , \quad ( \text{ for fixed } \rho, \ \tau_0, \ d_0 \text{ and } \psi_0 \ ) \quad .$$

In this run $\psi_0$ is fixed at $\psi_0 = 0$. This ensures that the starting point always lies along the curve where $f_3(\mathbf{x}(0)) = 0$; see Figure 6.10 for a schematic representation. In this run

$L$ increases until an orbit of desired length has been obtained. Figure 6.11 illustrates the result up to this stage.

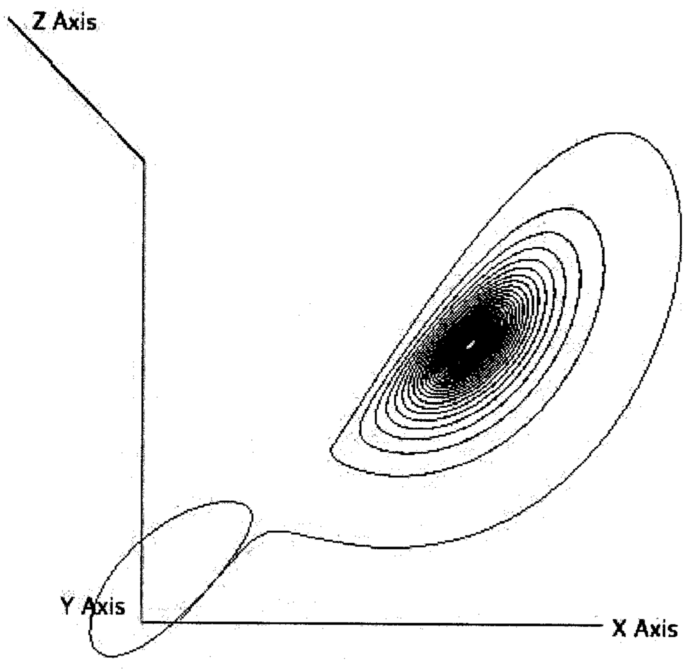


Figure 6.11: An orbit segment of length $L = 1000$ in the unstable manifold of the Lorenz system

**Run** 4:

In this run we keep $T$ fixed and let $d_0$ vary. Thus $X$ is the form

$$X = ( \mathbf{x}(\cdot) , d_0 , L , \tau_1 , d_1 ) \quad , \quad ( \text{for fixed } \rho, \ \tau_0, \ \psi_0 \text{ and } T ) \quad .$$

This run generates the unstable manifold. Zeroes of $\tau_1$ are also detected, which corresponds to the endpoint $\mathbf{x}(1)$ of the orbit passing through the stable eigenspace of the origin; see Figure 6.12 for part of the unstable manifold generated in this step.

Another possibility is that, instead of fixing $T$ in Run 4, we fix $L$ and free $T$ , so $X$ will be

$$X = ( \mathbf{x}(\cdot) , d_0 , T , \tau_1 , d_1 ) \quad , \quad ( \text{for fixed } \rho, \ \tau_0, \ \psi_0 \text{ and } L ) \quad .$$

Figure 6.13 and Figure 6.14 illustrate the results of this continuation, where the computation is done in the opposite continuation direction in Figure 6.14. Figure 6.15 shows the result when we merge the two figures above together.

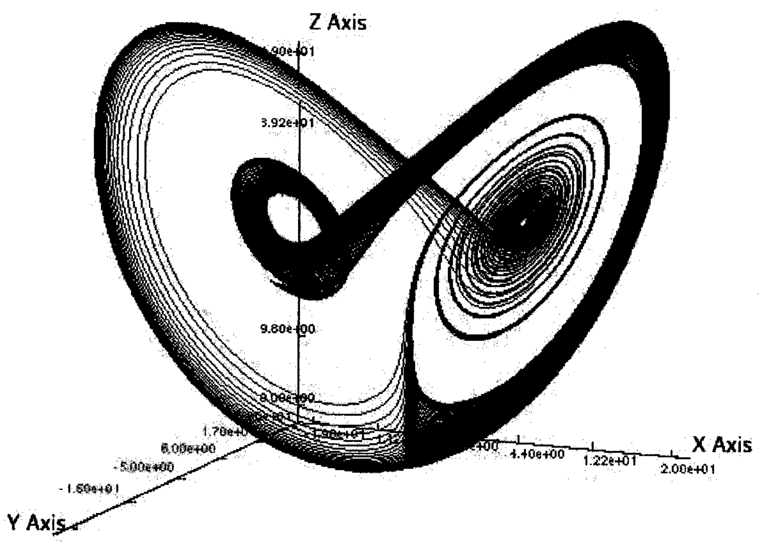


Figure 6.12: Part of the unstable manifold of the Lorenz system; $\rho$, $\tau_0$, $\psi_0$ and $T$ are fixed while $d_0$, $d_1$, $\tau_1$ and $L$ are variable, stopping after a specified number of continuation steps.

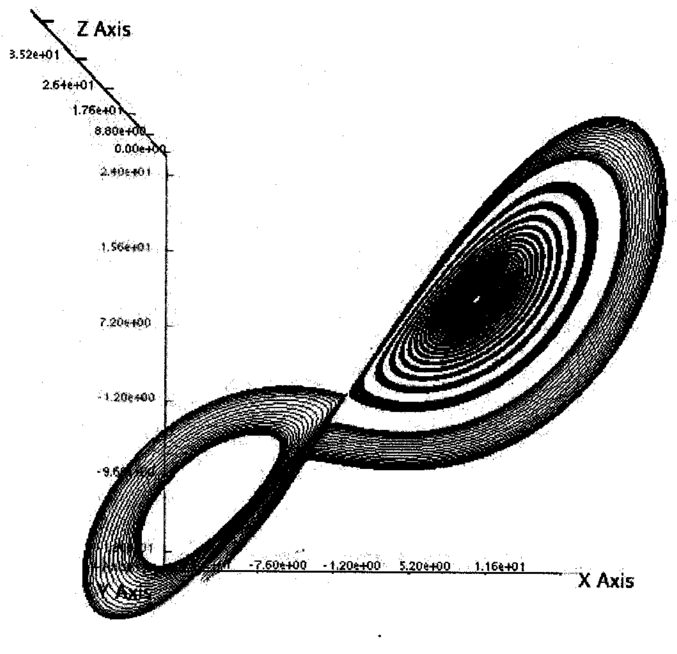


Figure 6.13: Part of the unstable manifold of the Lorenz system; $\rho$, $\tau_0$, $\psi_0$ and $L$ are fixed while $d_0$, $d_1$, $\tau_1$ and $T$ are variable, stopping after a specified number of continuation steps.

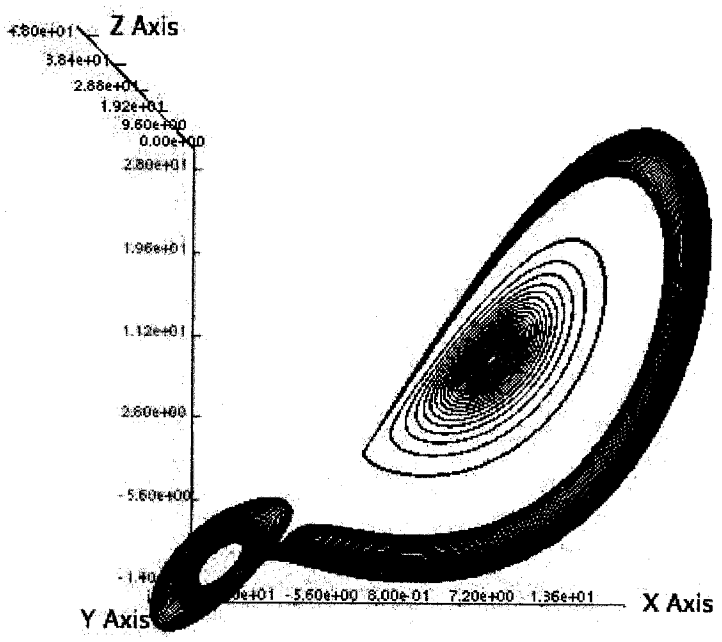


Figure 6.14: Part of the unstable manifold of the Lorenz system; $\rho$, $\tau_0$, $\psi_0$ and $L$ are fixed while $d_0$, $d_1$, $\tau_1$ and $T$ are variable, stopping after a specified number of continuation steps. Pseudo-arclength stepsize is in the opposite direction compared to Figure 6.13 .

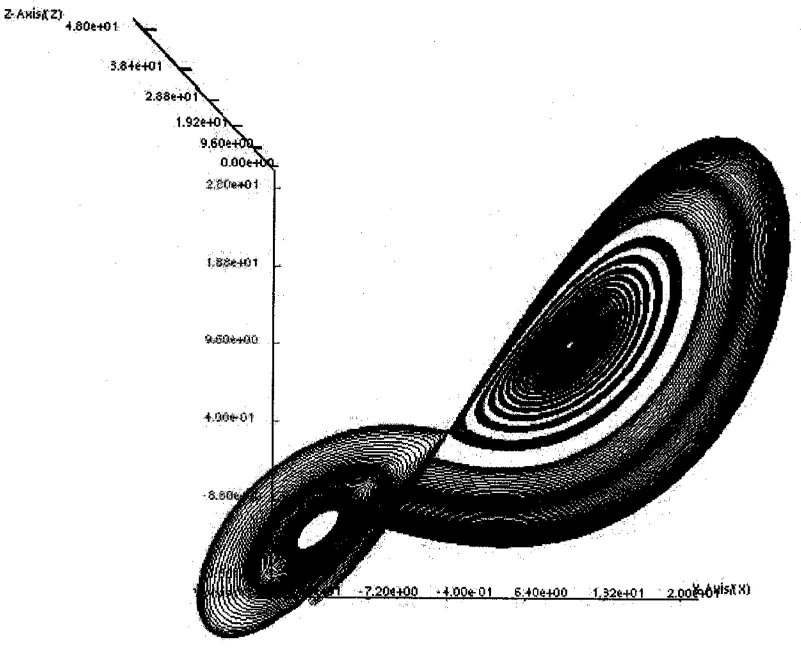


Figure 6.15: Part of the unstable manifold of the Lorenz system; This is the result when Figure 6.13 and Figure 6.14 are merged.

As mentioned, during the continuation of the unstable manifold we locate zeroes of $\tau_1$. Corresponding orbits have the property that the endpoint $\mathbf{x}(1)$ lies in $E_0^s$, the stable eigenspace of the origin. If, in addition, $||\mathbf{x}(1)||$ is small for such orbits then the orbit represents an approximate *heteroclinic orbit.* The following chapter addresses these heteroclinic connections in more detail.

Furthermore, computations can be done by varying $d_0$ over a *fundamental domain*, that is, a small range between successive detections of the basic heteroclinic orbit that spirals around the non-zero stationary points and connects to the origin; see [30] for more detail. Figure 6.16 contains 32 heteroclinic connections detected as zeroes of $\tau_1$ when $d_0$ varies over a fundamental domain.

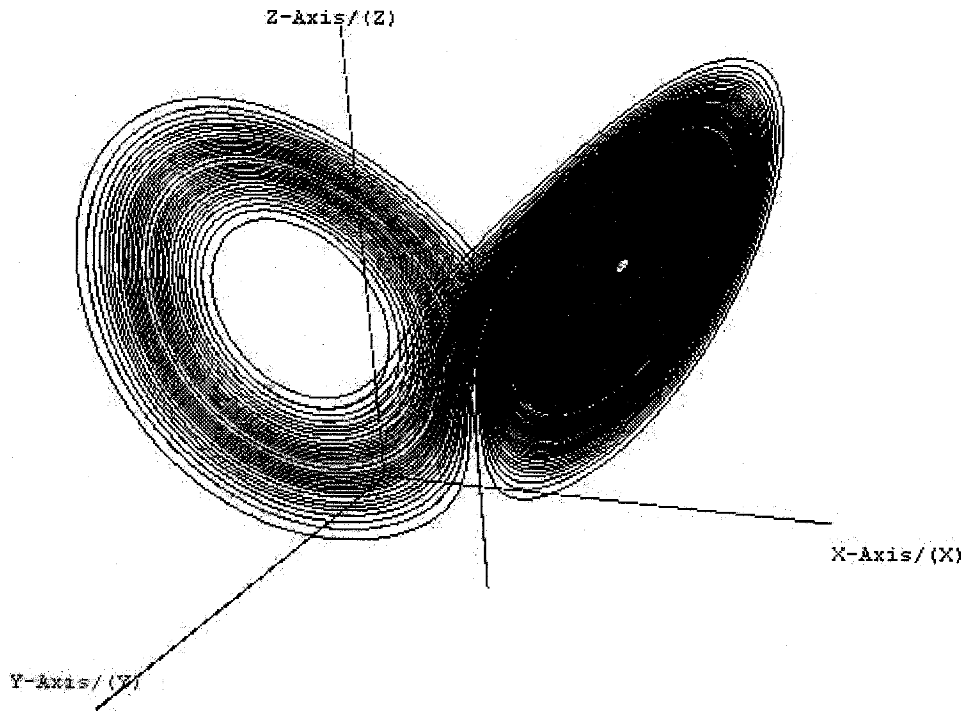


Figure 6.16: 32 heteroclinic connections in the Lorenz system

# Chapter 7

# Computation of Heteroclinic Connections

## 7.1 Introduction

In Wikipedia [78], it is stated that

> *In mathematics, in the phase portrait of a dynamical system, a heteroclinic orbit is a path in phase space which joins two different equilibrium points. If the equilibrium points at the start and end of the orbit are the same, the orbit is a homoclinic orbit.*

Heteroclinic orbits, which are orbits of infinite period, play an important role in nonlinear differential equations. They connect equilibria of a vector field. We now consider the computation of heteroclinic orbits that connect a non-zero stationary point to the origin in the Lorenz system.

## 7.2 Heteroclinic orbits for $\rho = 28.0$

The computation follows the procedure for computing the unstable manifold of the non-zero stationary point, as briefly summarized below.

The starting procedure is the same as for the unstable manifold of the non-zero stationary point. First, we do a continuation in the integration time $T$ (including other parameters as discussed in Section 6.3), where we start from an initial point along a particular direction in the unstable eigenspace $E^u_{sp}$ at distance $d_0$ from the non-zero stationary point. Thereafter, we perform two intermediate continuation steps, during the first of which an orbit family is continued until $\psi_0 = 0$, and during the second of which the length $L$ and the integration time $T$ are further increased. Then we fix the integration time $T$ and continue the orbit segment by varying $d_0$, $d_1$, $\tau_1$ and $L$. Thus, $F(X)$ takes the form

$$F(X) \equiv \begin{cases} \mathbf{x}'(t) - T\,\mathbf{f}(\,\mathbf{x}(t)\,,\,\rho\,) \\ <\mathbf{x}(0) - \mathbf{x}_{sp}\,,\,\mathbf{w}_0> \; -\tau_0 \\ <\mathbf{x}(1) - \mathbf{0}\,,\,\mathbf{w}_1> \; -\tau_1 \\ ||\,\mathbf{x}(0) - \mathbf{x}_{sp}\,|| - d_0 \\ ||\,\mathbf{x}(1) - \mathbf{0}\,|| - d_1 \\ T\,\int_0^1 ||\,\mathbf{f}(\mathbf{x}(s))\,||\,ds \; - L \\ f_3\,(\,\mathbf{x}(0)\,) - \psi_0 \end{cases} \tag{7.1}$$

with

$$X = (\,\mathbf{x}(\cdot)\,,\,d_0\,,\,L\,,\,\tau_1\,,\,d_1\,)\quad,\quad(\text{ for fixed } \rho,\ \tau_0,\ \psi_0 \text{ and } T\,)\quad.$$

During this process, the continuation passes through heteroclinic connections; detected as orbits for which $\tau_1 = 0$, with $||\mathbf{x}(1)||$ small. See Figure 7.1 for the computed manifold.

Orbits near heteroclinic connections spend much time near the origin. Therefore, since $T$ is fixed, heteroclinic orbits actually correspond to minima of the arclength $L$, as well as minima of the $L_2-$norm. Figure 7.2 shows the simplest heteroclinc orbit encountered.

We delete those orbits which are not heteroclinic connections from the solution file; see Figure 7.3 for the result. Figure 7.4 is the bifurcation diagram for fixed $T$ over a fundamental domain of $d_0$, indicating 32 approximate heteroclinic connections, which shows $||\cdot||$ versus radius $d_0$. The minima along the solution branch in Figure 7.4 correspond to heteroclinic connections. Each heteroclinic connection is also shown separately in 32 small figures at the end of this section.

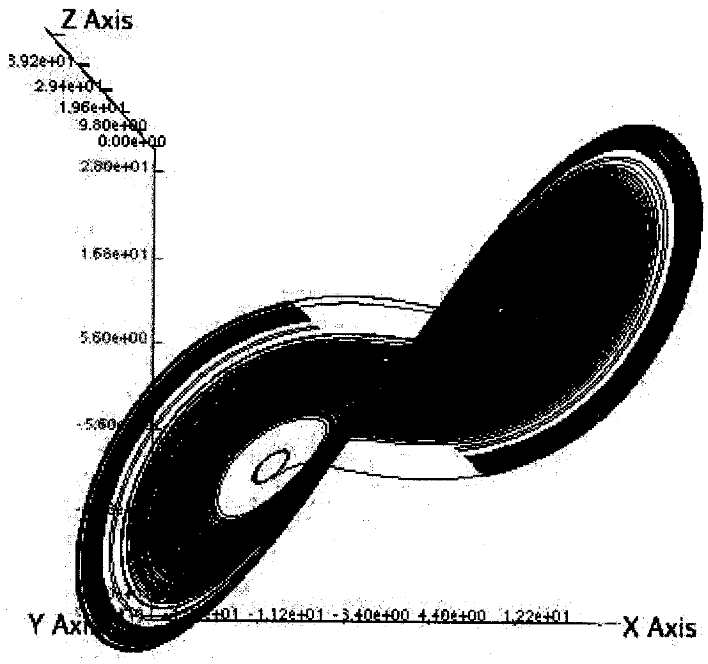


Figure 7.1: The manifold before deletion of non-heteroclinic orbits. Computed with $\rho$ , $\tau_0$, $\psi_0$ and $T$ fixed, $d_0$, $d_1$ , $\tau_1$ and $L$ variable.

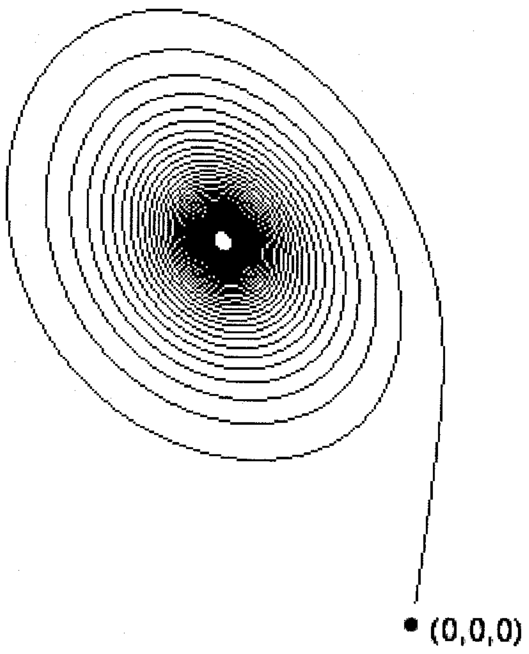


Figure 7.2: The simplest heteroclinic orbit in the Lorenz equations

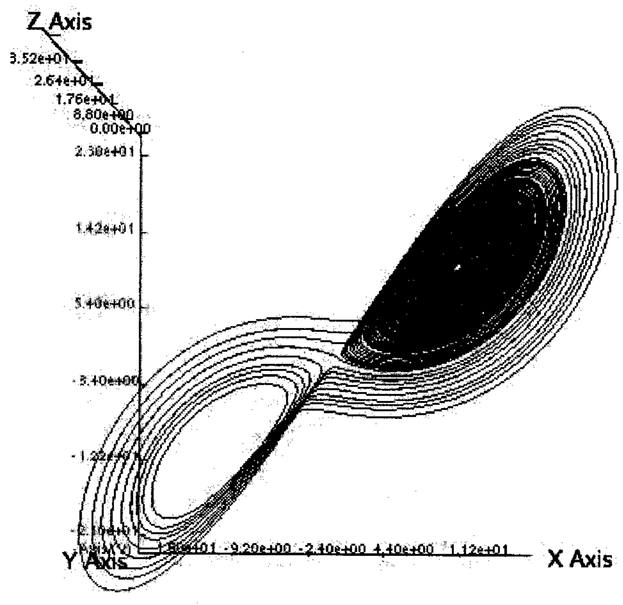


Figure 7.3: Heteroclinic connections that remain after deletion of all non-heteroclinic orbits

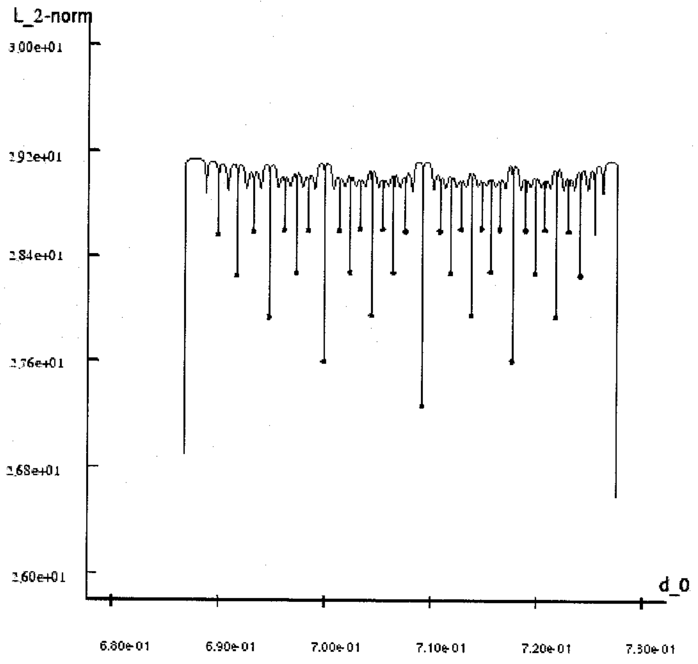


Figure 7.4: Bifurcation diagram marking 32 heteroclinic orbits, for $\rho = 28.0$ and fixed time $T$, and over a full fundamental domain of $d_0$. Local minima of the $L_2$ – norm correspond to (approximate) heteroclinic connections from $\mathbf{x}_{sp}^{+}$ to $\mathbf{0}$ .

## 7.3 Other computations with heteroclinic orbits

There are other useful computations involving heteroclinic orbits. For example, it is of interest to see how a heteroclinic orbit changes if the parameter $\rho$ is varied.

We can follow each computed heteroclinic orbit, with varying $\rho$, keeping $\tau_1$ equal to zero.

More precisely, in this case, $F(X)$ takes the form (7.1), where

$$X \;=\; (\; \mathbf{x}(\cdot)\;,\; \rho\;,\; d_0\;,\; T\;,\; L\;) \quad , \quad (\text{ with } \tau_0 = \tau_1 = \psi_0 = 0 \text{ and } d_1 \text{ ``small''}) \quad .$$

Suppose a particular heteroclinic orbit is chosen from the solution file that corresponds to Figure 7.3; see Figure 7.5 for a graphical illustration. Then we can follow this heteroclinic orbit with $\tau_1$ fixed at $\tau_1 = 0$, to see how it evolves for increasing $\rho$; see Figure 7.6. We can see that a butterfly structure appears.

A detailed discussion of heteroclinic connections in the Lorenz equations is given in [30].

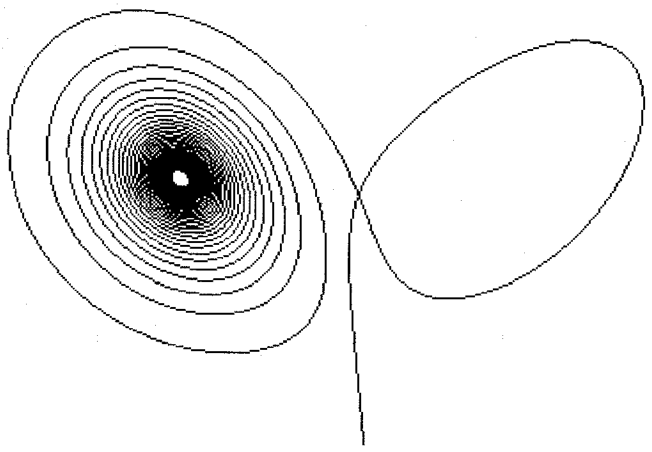


Figure 7.5: A heteroclinic orbit from the solution file corresponding to Figure 7.3

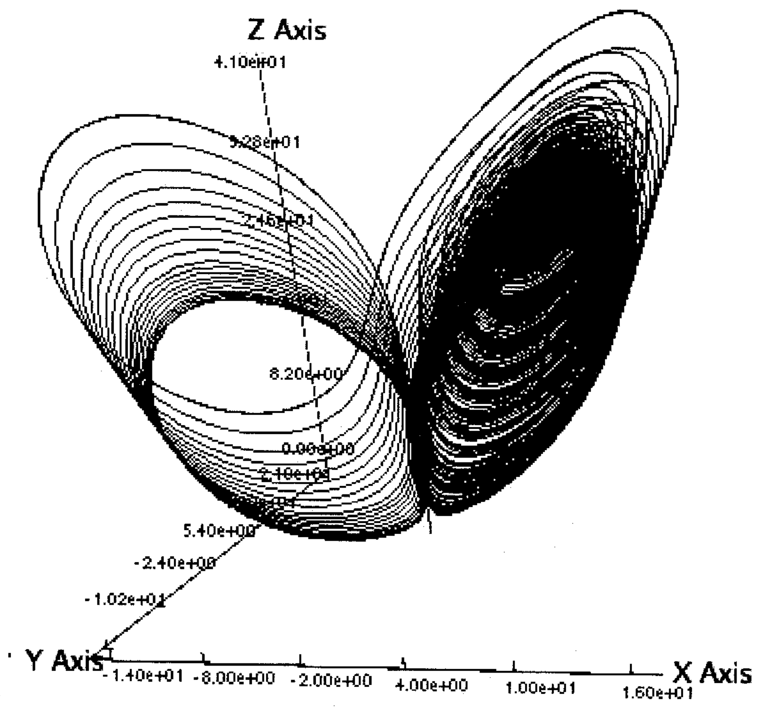
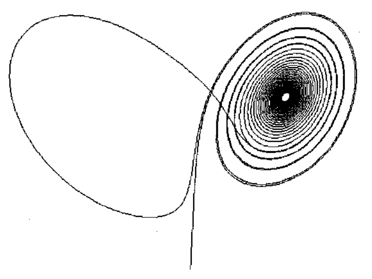
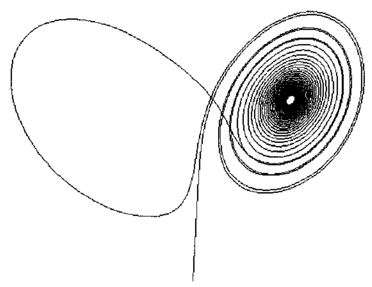
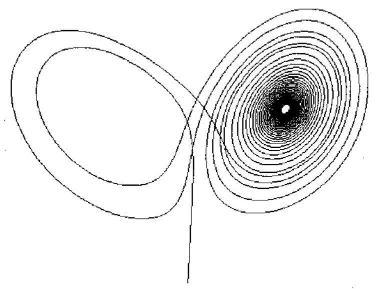
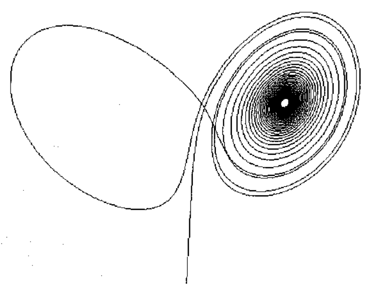


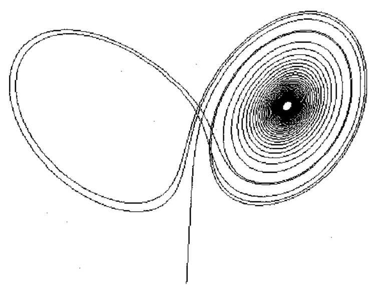
Figure 7.6: A family of heteroclinic connections for increasing $\rho$: the 16th heteroclinic orbit in figure 7.4 is used as starting orbit.

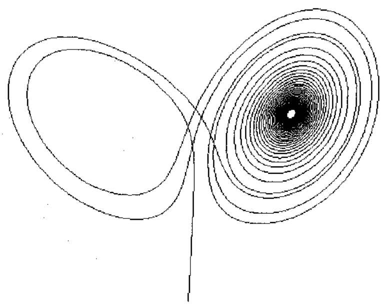

(a) Het_01

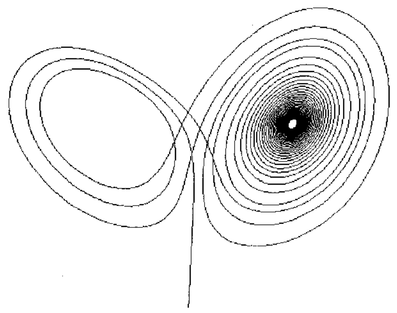

(b) Het_02


(c) Het_03


(d) Het_04


(e) Het_05


(f) Het_06


(g) Het_07

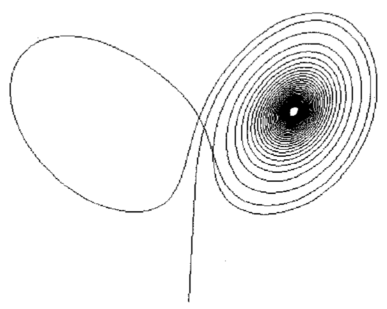
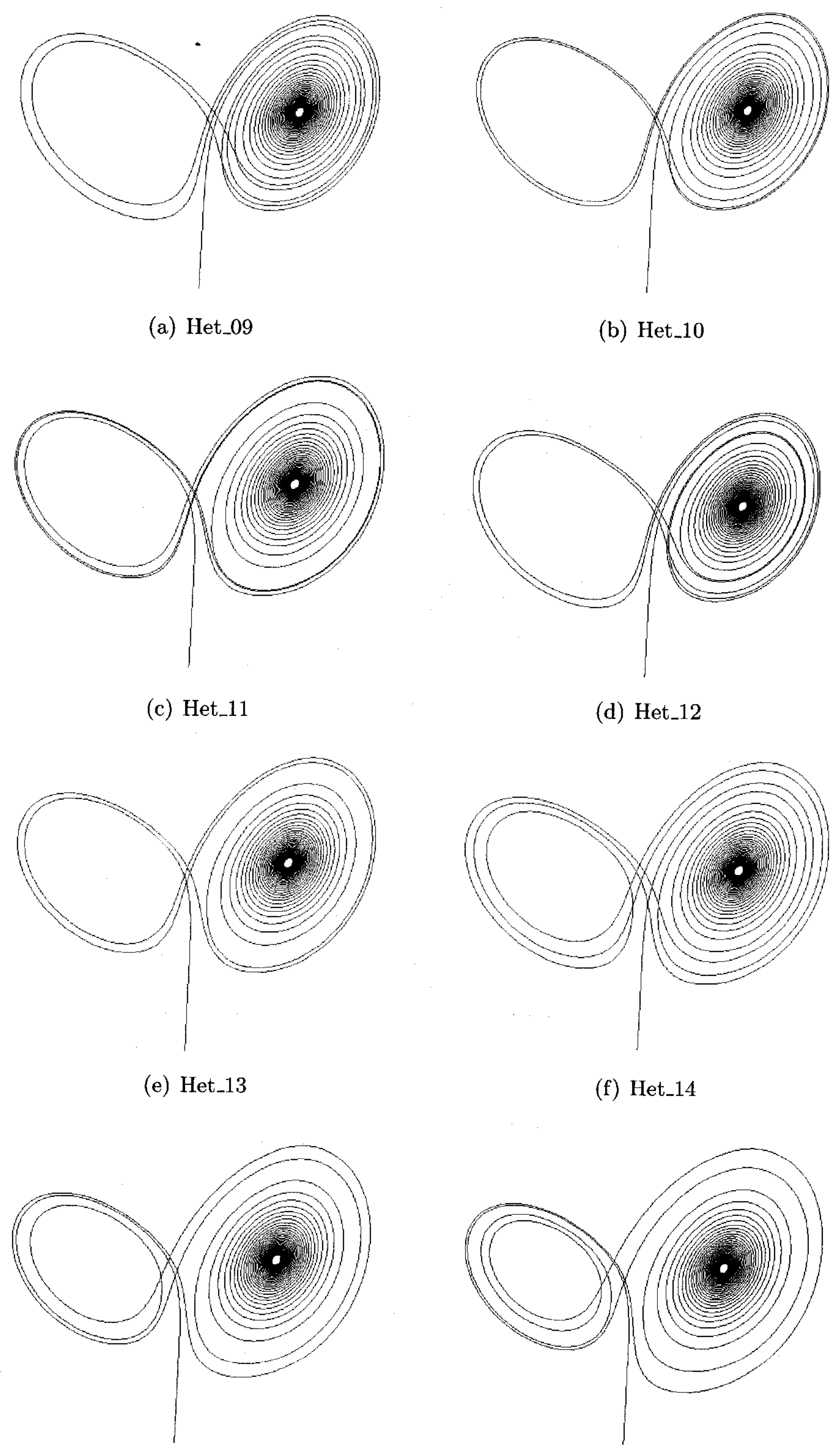

(h) Het_08

Figure 7.7: 32 heteroclinic connections over a fundamental domain of $d_0$: I



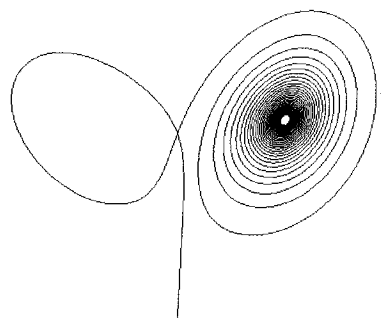
(a) Het_09 (b) Het_10

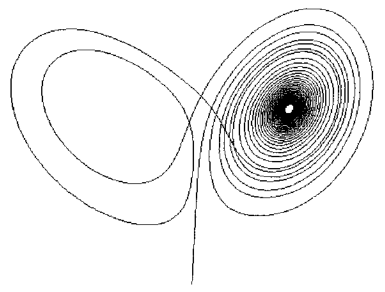
(c) Het_11 (d) Het_12

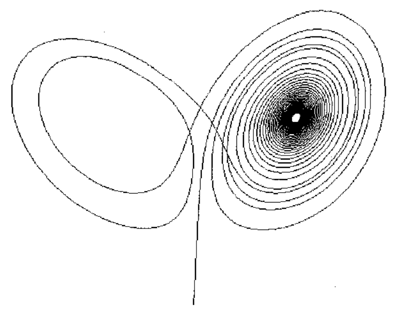
(e) Het_13 (f) Het_14

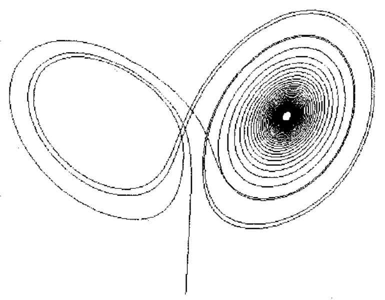
(g) Het_15 (h) Het_16

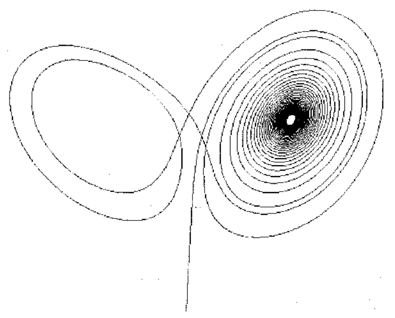
Figure 7.8: 32 heteroclinic connections over a fundamental domain of $d_0$: II

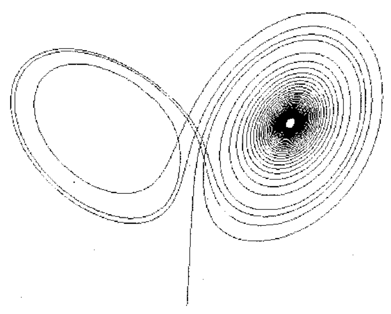

(a) Het_17

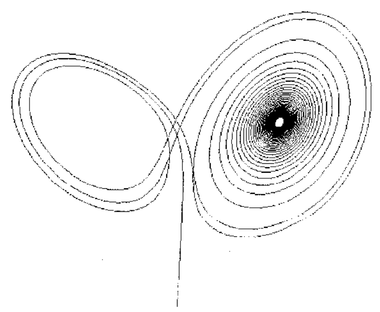

(b) Het_18


(c) Het_19


(d) Het_20


(e) Het_21


(f) Het_22


(g) Het_23

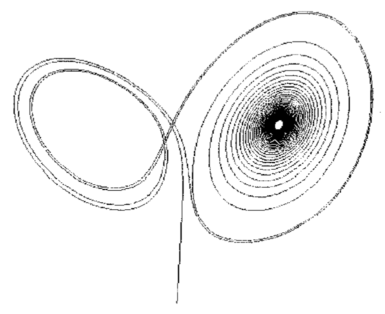
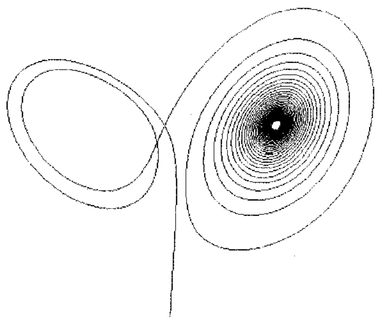
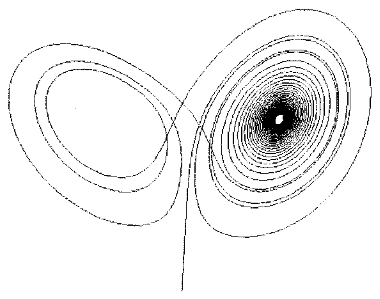

(h) Het_24

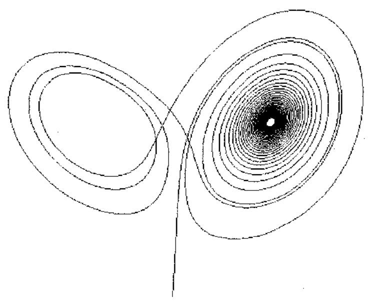
Figure 7.9: 32 heteroclinic connections over a fundamental domain of $d_0$: III



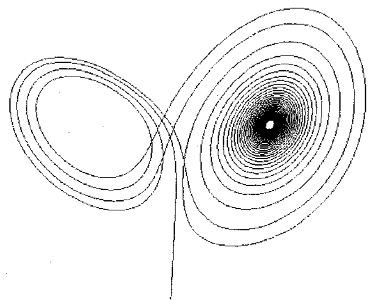
(a) Het_25



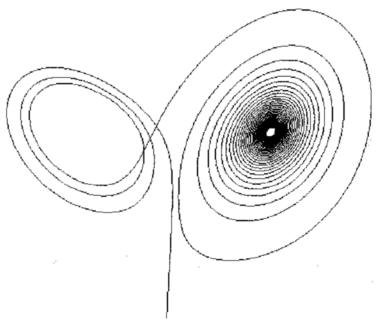
(b) Het_26



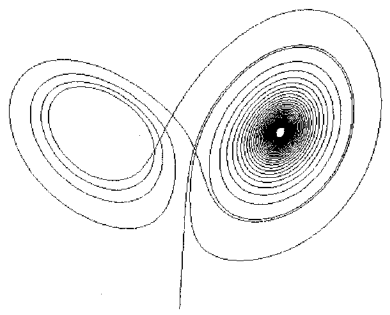
(c) Het_27



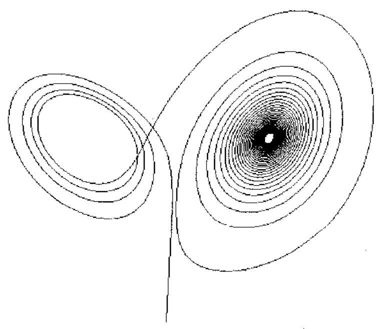
(d) Het_28



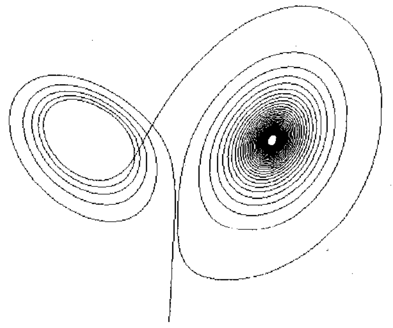
(e) Het_29



(f) Het_30



(g) Het_31



(h) Het_32

Figure 7.10: 32 heteroclinic connections over a fundamental domain of $d_0$: IV

# Chapter 8

# Visualization

The computational results give large amounts of data. In order to better understand the nature of the computed manifolds, a graphics tool has been developed to display the families of computed curves as surfaces. This is done by generating a triangulation that approximately represents the manifold. Functions such as resetting, moving, scaling and rotating are also provided.

## 8.1 Objectives

There are many papers that discuss computations for the Lorenz System; see [61, 60, 44, 71]. Most of these focus on the rough shape of the Lorenz manifolds, as it is hard to explain exactly how the manifolds really look like. This leads to the idea to develop a particular graphics tool to address this problem. It is also useful to view different objects that are related to the manifolds, such as homoclinic connections, heteroclinic connections, periodic orbits, *etc.*

In general, the graphics tool should also be able to adapt to different solutions files; and not only for data files generated by AUTO.

## 8.2 Graphics Interface

C/C++ is the language used for writing the graphics tool because of its flexibility, efficiency and availability on most platforms. There are a number of existing C/C++ libraries, which make the implementation of the tool easier.

At present many methods can be used to plot 3D graphics, but some of them are not portable, such as MicroSoft Direct 3D, which is only for Windows.

OpenGL is a C/C++ 3D graphics interface that is very flexible and portable to most operating systems, [72, 70, 47, 79]. Also, it has some functions that can accelerate the interpolation of curves, which we require for plotting manifolds. Nevertheless, OpenGL does not provide high-level commands for describing models of three-dimensional objects; the model should start from a set of primitives — points, lines, and polygons. There exists a high-level, object-oriented toolkit, Open Inventor, which is built on the top of OpenGL, and is available separately for many implementations of OpenGL. The software package PLAUT04 ( which is included in the AUTO-07p version of AUTO ), deploys Open Inventor for visualizing graphics. We also enhanced some functions of PLAUT04.

We need a criterion to distinguish visually between trajectories, *e.g.,* using various colors to distinguish trajectories or some quantity that varies along trajectories. In this tool, we adopt the colormap with 64 different colors as in Matlab [57].

## 8.3 Problem specific implementation

As mentioned in Chapter 4, AUTO-07p is the software we used to compute the manifolds of the Lorenz system. The approach is to follow an orbit segment, while the starting point is allowed to vary along an ellipse or line segment in the stable or unstable eigenspace centered at a stationary point. Pseudo-arclength continuation is then applied, and several boundary conditions are imposed. When using the collocation method, we keep all mesh points fixed during any particular run. The number of mesh points can be changed when restarting computations from a selected solution. Thus, it is possible to have a different number of points for two adjacent orbits; see Figure 8.1.

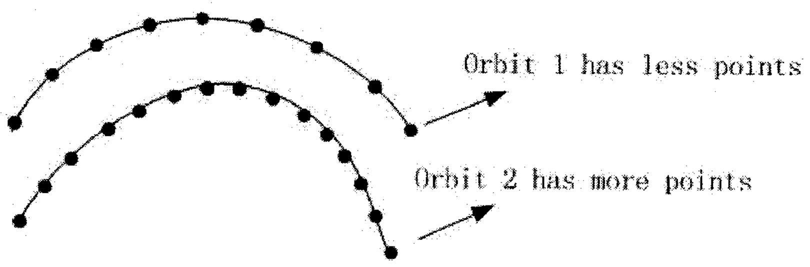


Figure 8.1: Different number of points on adjacent trajectories

A problem is that the points along a trajectory are not uniformly distributed; one point might be quite far away from its corresponding point on an adjacent trajectory, resulting in inaccurate triangles; for example, see Figure 8.2.

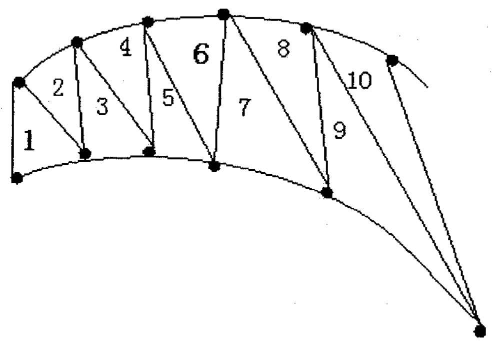


Figure 8.2: Irregular triangle formation along trajectories on which points are not uniformly distributed

In addition, it is possible that the solution points generated by AUTO are not dense enough, *i.e.*, the stepsize used in the computation is not small. The method to solve these problems is *interpolation*. It is a method for estimating values that lie between known values. For more details, see [14, 64].

In the case of the Lorenz system the stepsize is generaly small and solution points generated by AUTO are usually dense enough, so that we do not use interpolation here.

## 8.4 The triangulation algorithm

We illustrate the implementation using the Lorenz system. In principle, the algorithm applies to other applications.

Since computations are done by numerical continuation method in AUTO, every two adjacent orbits are "close" to each other, so triangulation [52] can be applied to form surfaces between these trajectories. As explained in the Section 8.3, adjacent orbits may or may not have the same number of points. The algorithm applies to both cases.

### 8.4.1 Orbits with the same number of points

To illustrate the algorithm, first suppose orbit A and orbit B each has 5 points. Furthermore, assume that orbit A starts from $a_1$ and ends in $a_5$, and orbit B starts from $b_1$ and ends in $b_5$.

First we construct two triangles, namely, triangle $< a_1 \cdot b_1 \cdot a_2 >$ and triangle $< a_1 \cdot b_2 \cdot b_1 >$; see Figure 8.3.

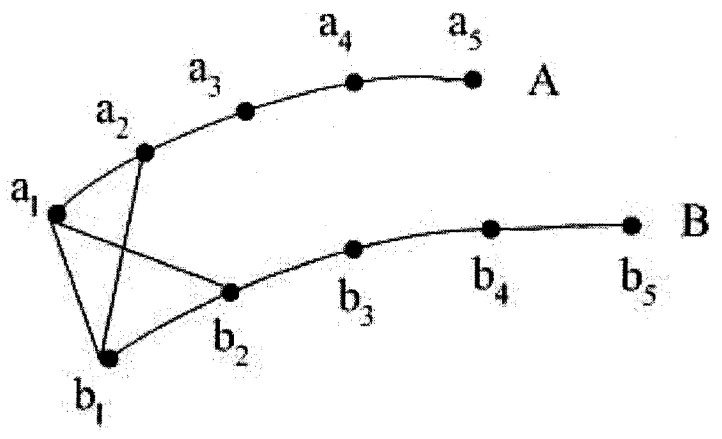


Figure 8.3: Triangulation between two adjacent orbits with the same number of points: I

Then we compare the two angles, $\angle\ a_1 \rightarrow b_2 \rightarrow b_1$ with $\angle\ a_1 \rightarrow a_2 \rightarrow b_1$. If angle $\angle\ a_1 \rightarrow a_2 \rightarrow b_1$ less than angle $\angle\ a_1 \rightarrow b_2 \rightarrow b_1$, then discard the triangle $< a_1 \cdot b_1 \cdot a_2 >$ and keep the triangle $< a_1 \cdot b_2 \cdot b_1 >$. Otherwise discard the triangle $< a_1 \cdot b_2 \cdot b_1 >$ and keep the triangle $< a_1 \cdot b_1 \cdot a_2 >$; see Figure 8.4.

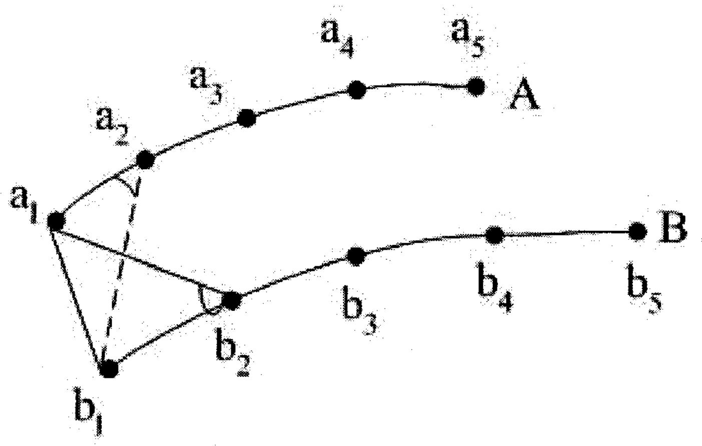


Figure 8.4: Triangulation between two adjacent orbits with the same number of points: II

In our case, since the triangle $< a_1 \cdot b_2 \cdot b_1 >$ has been kept, we use line $a_1b_2$ as the baseline to construct another two triangles: $< a_1 \cdot b_3 \cdot b_2 >$ and $< a_1 \cdot a_2 \cdot b_2 >$.

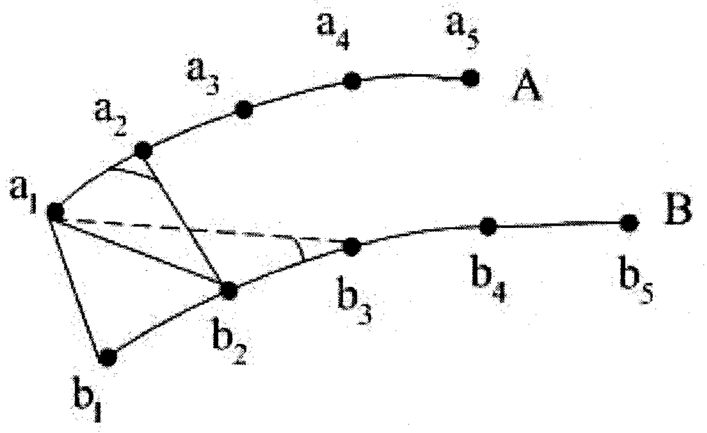


Figure 8.5: Triangulation between two adjacent orbits with the same number of points: III

Again we compare angle $\angle\, a_1 \rightarrow a_2 \rightarrow b_2$ with angle $\angle\, a_1 \rightarrow b_3 \rightarrow b_2$, discard the triangle with the smaller angle and keep the other triangle. The same procedure continues until the end. In this way, the triangles constructed are generally in good shape. In addition, the algorithm is very fast since we do not need computation during the construction; see Figure 8.6 for all triangles constructed.

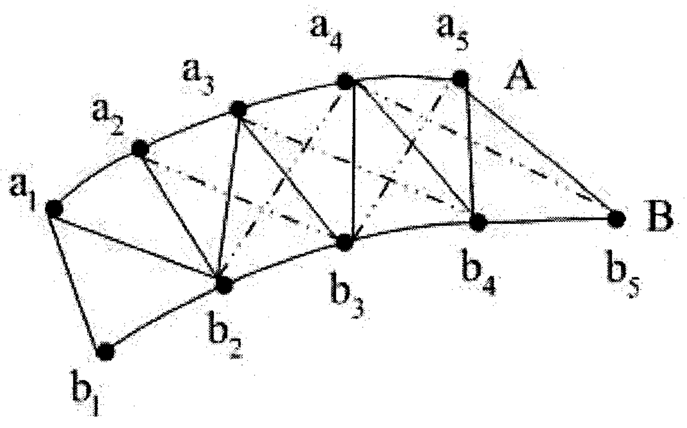


Figure 8.6: Triangulation between two adjacent orbits with the same number of points: IV

### 8.4.2 Orbits with a different number of points

Triangle construction for orbits with a different number of points follows the same pattern. If orbit A reaches the endpoint while orbit B still has more solution points then we simply connect each remaining point on orbit B to the endpoint on orbit A; see Figure 8.7. Experiments done for the Lorenz equations shows that this approach is generally more than adequate for AUTO-generated data.

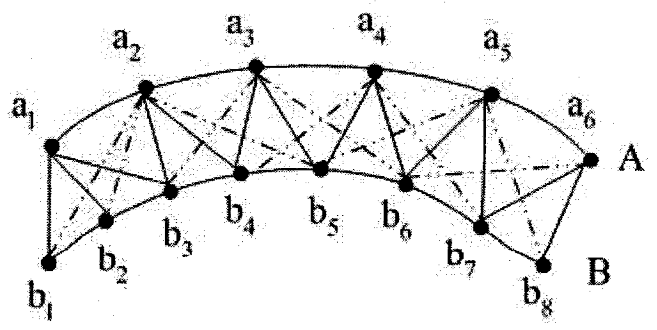


Figure 8.7: Triangulation between two adjacent orbits with a different number of points

In case of a large solution file, it is not a good idea to connect every point on one orbit to its corresponding point on the other orbit. The number of mesh intervals used in the computations is mainly determined by the required accuracy. A better approach is to choose some points only, *e.g.*, plotting every 10th point on each orbit and forming triangles between them. Another even more efficient approach is to plot only some trajectories. Often there is no reason to form triangles between all orbits. All approaches mentioned above can be applied to the solution files.

Below are figures for the Lorenz manifold plotted with the new graphical tool VMD. Figure 8.8 shows the same view as plotted by PLAUT04. Figure 8.9 shows the triangulation and Figure 8.10 shows a local blow-up of the triangulation.

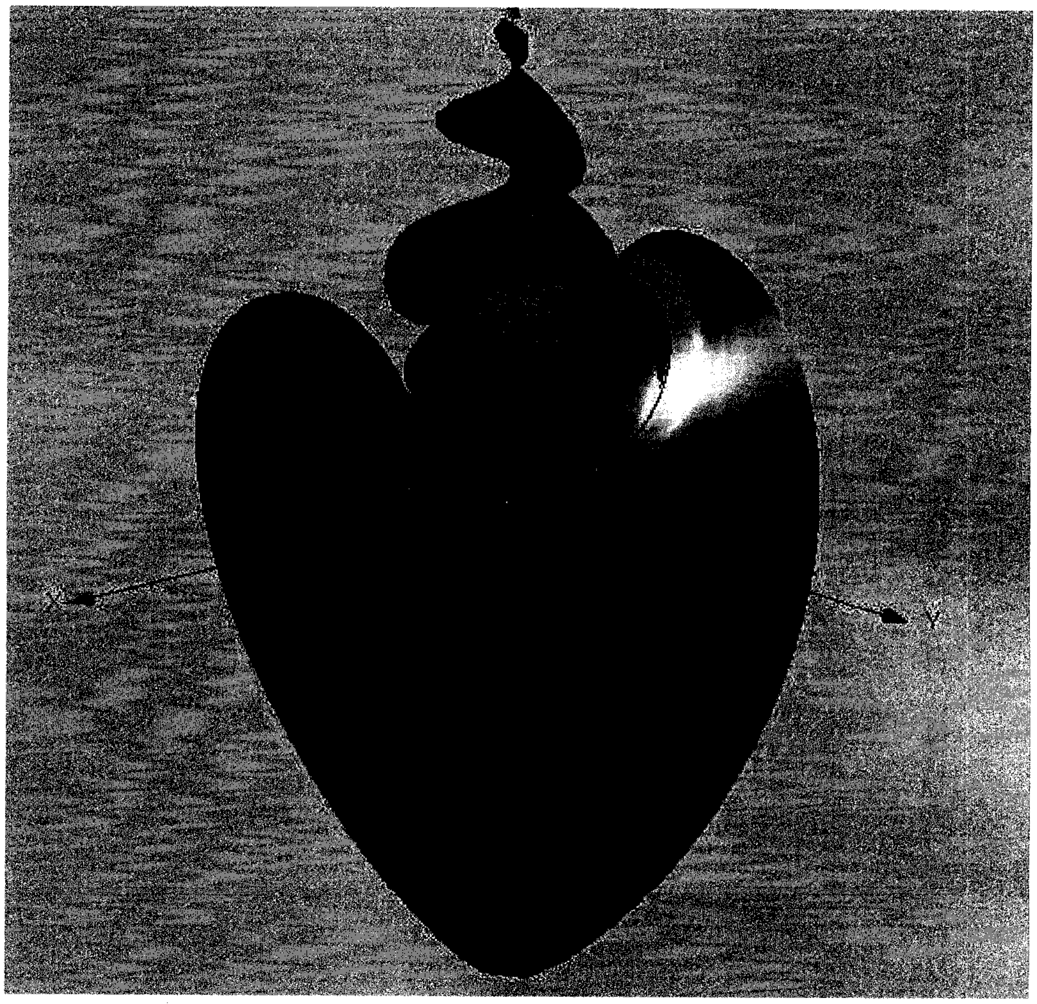


Figure 8.8: The Lorenz manifold plotted by VMD

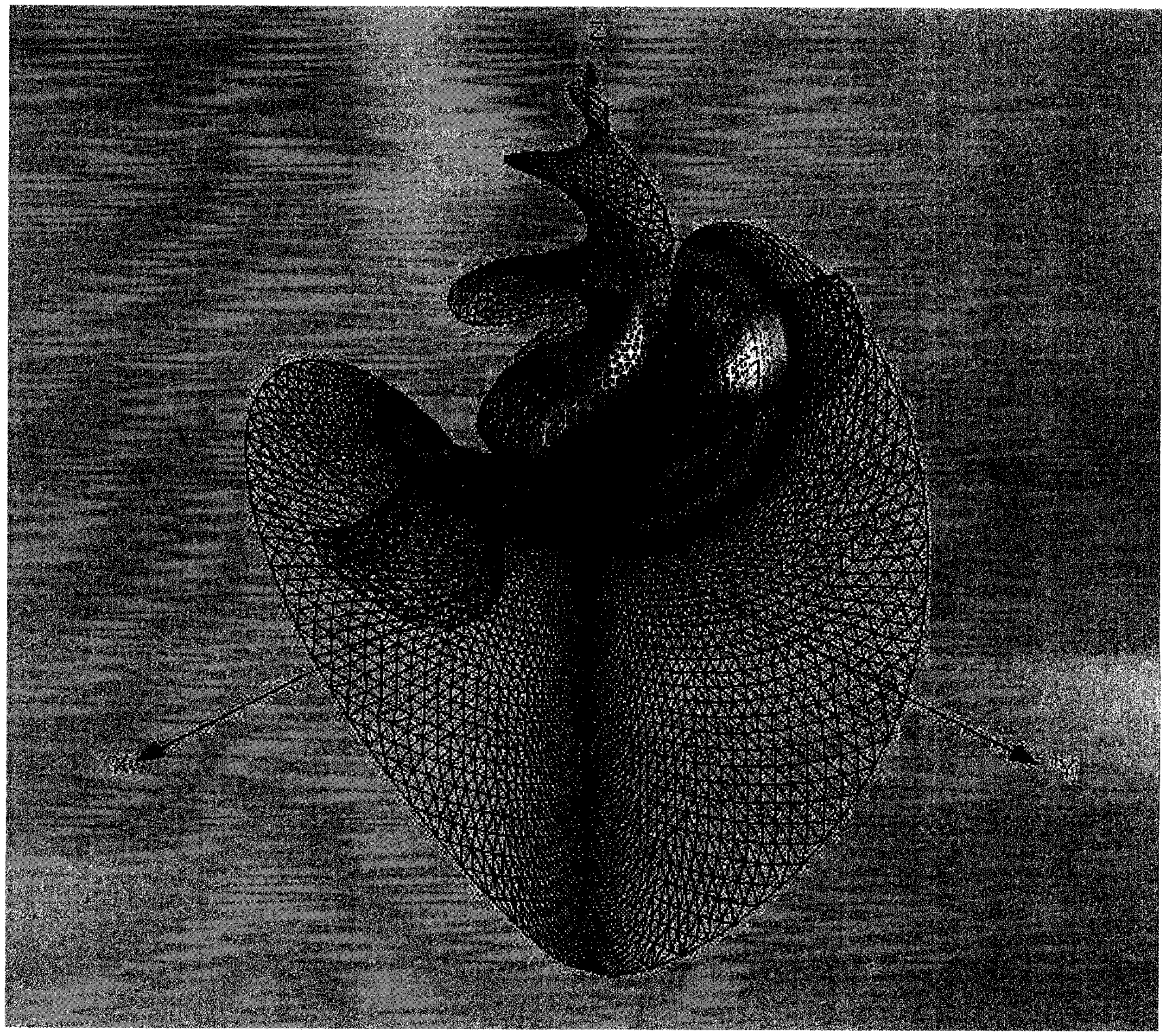


Figure 8.9: The Lorenz manifold plotted by VMD: showing the triangulation

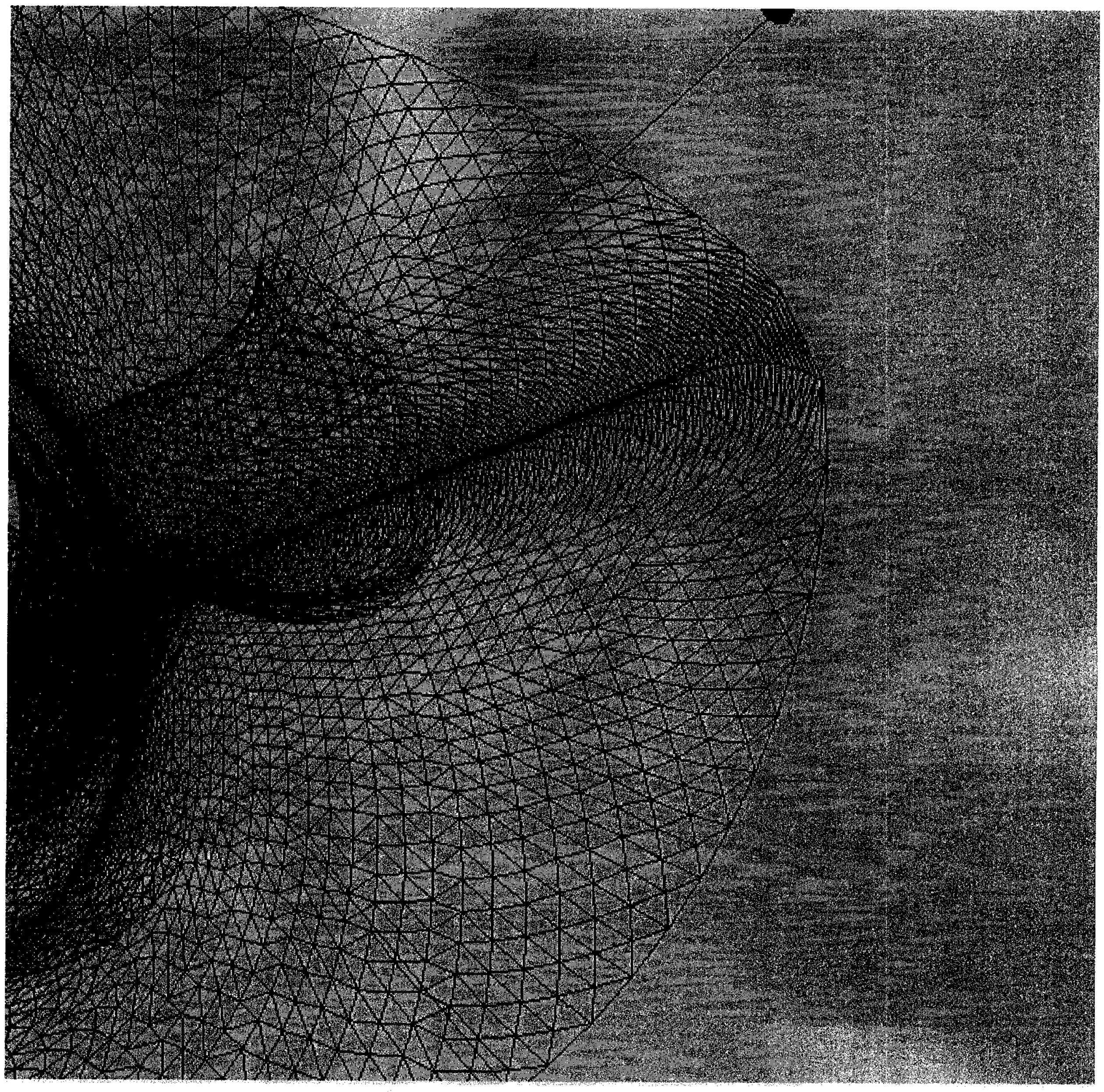


Figure 8.10: The Lorenz manifold plotted by VMD: local view of the triangulation

# Chapter 9

# Conclusions and Discussion

## 9.1 Conclusions

Numerical continuation with AUTO was successfully used in this thesis for the computation of stable and unstable manifolds in the Lorenz equations. Heteroclinic connections from the non-zero stationary point to the origin of the Lorenz system were detected. Orbits and their corresponding bifurcation diagrams are depicted to give us a better understanding of the manifolds and the heteroclinic connections.

A new data visualization program, VMD, was developed for visualizing AUTO datasets, and especially for observing manifolds. Its features, such as portability, simplicity, zooming capabilities *etc.*, make it useful for AUTO users. The implementation of triangulation gives users a more accurate view of the manifolds in three dimensional space. VMD is a general purpose program for problems related to manifolds. Its processing speed for large datasets is reasonable.

The continuation and bifurcation software package AUTO-07p, which is an updated version of AUTO97, was used in this thesis. AUTO-07p incorporates a Python interface, which enhances its power. Some changes and updates made in AUTO are detailed in the Appendices.

## 9.2 Future development

Further development could include combining the graphics package PLAUT04 with the most recent graphics tool VMD. Since VMD is implemented separately and mainly focuses on visualization of manifolds, it will be a good idea to add the functionalities of VMD to PLAUT04, thus enhancing its power and avoiding overlap between the two programs.

There is another software package, *the Visualization ToolKit*, VTK in short, which is an open source software system for 3D computer graphics, image processing and visualization. It can be directly integrated through Python. The computation power, storage capability and other functions provided by VTK are much more comprehensive than those of Open Inventor. So future version of AUTO graphics could focus on the use of the new VTK package for achieving better views of AUTO data.

The current algorithm for computing manifolds for the Lorenz system is very accurate and robust. A general built-in capability for computing manifolds with AUTO may be considered in the future.

# Bibliography

[1] R. H. Abraham and C. D. Shaw. "Dynamics–The Geometry of Behavior. Part three: global behavior." Aerial Press, Santa Cruz, 1985.

[2] R. K. Alexander. "Stability of Runge-Kutta Methods for Stiff Ordinary Differential Equations." SIAM Journal on Numerical Analysis, Vol. 31, pp. 1147-1168, 1994.

[3] E. L. Allgower and K. Georg. "Numerical Path Following." In P. G. Ciarlet and J. L. Lions, editors, Handbook of Numerical Analysis, North Holland Publishing, Vol. 5, 1996.

[4] P. Angel and M. C. Rivara. "Mesh Refinement Based on The 8-Tetrahedra Longest-edge Partition", available by HTTP from http://www.andrew.-cmu.edu/user/sowen/abstracts/P1977.html, 2th International Meshing Roundtable, Sandia National Laboratories, pp. 67-78, 2003.

[5] U. Ascher, J. Christiansen, and R.D. Russell. "Collocation Software for Boundary-Value ODEs." ACM Trans. Math. Software, Vol. 7, pp. 209-222, 1981.

[6] U. Ascher, J. Christiansen, and R. D. Russell. "COLSYS: Collocation Software for Boundary-Value ODEs." ACM Trans. Math. Software, Vol. 7, pp. 223-229, 1981.

[7] U. Ascher and G. Bader. "Stability of Collocation at Gaussian Points." SIAM J. Numer Anal., Vol. 23, pp. 412-422, 1986.

[8] A. Back, J. Guckenheimer, M. R. Myers, F. J. Wicklin, and P. A. Worfolk. "DsTool: Computer Assisted Exploration of Dynamical Systems." Notices Amer. Math. Soc., Vol. 39(4), pp. 303-309, 1992.

[9] R. E. Bank. A. H. Sherman, and A. Weiser. “Refinement Algorithms and Data Structures for Regular Local Mesh Refinement.” Scientific Computing, Brussels, IMACS/North Holland, Netherland, pp. 3-17, 1983.

[10] R. E. Bank and R. Kent Smith. “Mesh Smoothing Using a Posteriori Error Estimates.” SIAM J. Numer. Anal., Vol. 34, pp. 979-997, 1997.

[11] W. J. Beyn, A. Champneys, E. J. Doedel, W. Govaerts, B. Sandstede, and Yu. A. Kuznetov. “Numerical Continuation and Computation of Normal Forms.” In B. Fiedler, editor, ”Handbook of Dynamical Systems.” Elsevier Science, Vol. 2, pp. 149-219, 2001.

[12] B. Bialecki, G. Fairweather, and K. Bennett. “Fast Direct Solvers for Piecewise Hermite Bicubic Orthogonal Spline Collocation Equations.” SIAM J. Numer. Anal.,29, 1992.

[13] C. de Boor and B. Swartz. “Collocation at Gaussian points.” SIAM J. Numer.Anal., Vol. 10, pp. 582-606, 1973.

[14] C. de Boor. “A Practical Guide to Splines.” Springer, Berlin, Heidelberg, 1978.

[15] C. Bozzuto. “Machine Learning: Fractals and Dynamics.” available by HTTP from http://www.cs.brandeis.edu/ cs113/classprojects/ smoate/cs113/fractals _and_dynamics.html , 2002.

[16] T. N. Chan. “Numerical Bifurcation Analysis of Simple Dynamical Systems.” Master Thesis, Computer Science Department, Concordia University, 1983.

[17] Coin3D Package, available by HTTP from http://www.coin3d.org/doc/.

[18] CONTENT: A multiplatform environment for continuation and bifurcation analysis of dynamical systems. Developed by Y. A. Kuznetsov and V. V. Levitin, Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, 1997.

[19] DDE-BIFTOOL v.2.00: A Matlab package for bifurcation analysis of delay differential equations. Developed by K. Engelborghs, Technical Report TW-330, Department of Computer Science, University of Leuven, Belgium, 2001.

[20] M. Dellnitz and A. Hohmann. "The Computation of Unstable Manifolds Using Subdivision and Continuation." In H.W. Broer et al. (eds.), Progress in Nonlinear Differential Equations and Their Applications , Birkhäuser Verlag, Basel / Switzerland, Vol. 19, pp. 449-459, 1996.

[21] A. Deprit, and J. Henrard. "Construction of Orbits Asymptotic to a Periodic Orbit." Astron. J., Vol. 74, pp. 308-316, 1969.

[22] E. J. Doedel. "Finite Difference Collocation Methods for Nonlinear Two Point Boundary Value Problems." SIAM Journal on Numerical Analysis, Vol. 16, No. 2, pp. 173-185, 1979.

[23] E. J. Doedel. "AUTO: a program for the automatic bifurcation analysis of autonomous systems." In Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980), Cong. Num. Vol. 30, pp. 265-284, 1981.

[24] E. J. Doedel, H. B. Keller, and J. P. Kernévez. "Numerical Analysis and Control of Bifurcation Problems (I) Bifurcation in Finite Dimensions." International Journal of Bifurcation and Chaos, Vol. 1(3), pp. 493-520, 1991.

[25] E. J. Doedel, H. B. Keller, and J. P. Kernévez. "Numerical Analysis and Control of Bifurcation Problems (II) Bifurcation in Infinite Dimensions." International Journal of Bifurcation and Chaos, Vol. 1(4), pp. 745-772, 1991.

[26] AUTO97: Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont), developed by E. J. Doedel et al., available by HTTP from http://cmvl.cs.concordia.ca/auto/ .

[27] AUTO2000: Continuation and Bifurcation Software for Ordinary Differential Equations, a C language version of AUTO, developed by E. J. Doedel et al., available by HTTP from http://cmvl.cs.concordia.ca/auto/ .

[28] AUTO-07p: the successor to both AUTO97 and AUTO2000, developed by E. J. Doedel et al., available by HTTP from http://cmvl.cs.concordia.ca/auto/ .

[29] E. J. Doedel. "Numerical Analysis of Nonlinear Equations, Lecture Notes." available by HTTP from http://cmvl.cs.concordia.ca/doedel.html .

[30] E. J. Doedel, B. Krauskopf and H. M. Osinga. "Global bifurcations of the Lorenz manifold." Nonlinearity 19, pp. 2947-2972, 2006.

[31] DsTool: Computer assisted exploration of dynamical systems, developed by M. Myers, R. Wicklin, P. Worfolk and A. Back, available by HTTP from http://www.cam.cornell.edu/~gucken/dstool , 1992.

[32] F. J. Elmer. "The Pendulum Lab.", available by HTTP from http://monet.-unibas.ch/ elmer/pendulum/bif.htm , 1998.

[33] J. P. England. "Advances in computing global invariant manifolds." PhD thesis, Engineering Mathematics, University of Bristol, Chapter 6, 2005.

[34] "Fedora3 Linux Online Help." available by HTTP from http://fedora.redhat.com, Fedora3 manual, 2004.

[35] M. J. Friedman and E. J. Doedel. "Numerical Computation and Continuation of Invariant Manifolds Connecting Fixed Points." SIAM J. Numer. Anal. Vol. 28(3), pp. 789-808, 1991.

[36] D. Goldberg. "What Every Computer Scientist Should Know About Floating-Point Arithmetic." Issue of Computing Surveys, Association for Computing Machinery, Inc., 1991.

[37] J. Guckenheimer and P. Holmes. "Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields." second edition, Springer-Verlag, New York, 1986.

[38] J. Guckenheimer. "Dimension Estimates for Attractors." In Fluids and plasmas: geometry and dynamics (Boulder, Colo., 1983), Amer. Math. Soc.,Providence, RI, pp. 357-367, 1984.

[39] J. Guckenheimer and P. Worfolk. "Dynamical Systems: Some Computational Problems, in Bifurcations and Periodic Orbits of Vector Fields." D. Schlomiuk, ed. pp. 241-278, Kluwer 1993.

[40] J. Guckenheimer. "Numerical Analysis of Dynamical Systems." Amer. Math. Soc., Providence, RI, 1999.

[41] J. Hale, and H. Koçak. "Dynamics and Bifurcations." Springer-Verlag, New York, 1991.

[42] HiSee: A visualization tool for visualizing high-dimensional datasets, developed by S. Hotton and J. Yoshimi, available by HTTP from http://hisee.sourceforge.net/ .

[43] M. E. Johnson, M. S. Jolly, and I. G. Kevrekidis. "Two-dimensional invariant manifolds and global bifurcations: some approximation and visualization studies." Numerical Algorithms, Springer Netherlands, Vol. 14, Numbers 1-3 / April, pp. 125-140, 1997.

[44] N. D. Kazarinoff and R. Seydel. "Bifurcations and period doubling in E. N. Lorenz's symmetric fourth order system." Physical Review A Vol. 34, pp. 3387-3392, 1986.

[45] H. B. Keller. "Numerical solution of bifurcation and nonlinear eigenvalue problems." Applications of Bifurcation Theory, ed. Rabinowitz, P. H. Academic Press, pp. 359-384, 1977.

[46] H. B. Keller. "Lectures on numerical methods in bifurcation problems." Notes by Nandakumaran, A. K. & Ramaswamy, M., Indian Institute of Science, Bangalore. Springer-Verlag, 1987. Academic Press, pp. 359-384, 1977.

[47] M. J. Kilgard. "OpenGL Programming for the X Window System." Addison-Wesley, 1996.

[48] B. Krauskopf and H. M. Osinga. "Two-dimensional global manifolds of vector fields." *Chaos* Vol. 9(3), pp. 768-774, 1999.

[49] B. Krauskopf and H. M. Osinga. "Computing geodesic level sets on global (un)stable manifolds of vector fields." SIAM J. Appl. Dyn. Sys. Vol. 4(2), pp. 546-569, 2003.

[50] B. Krauskopf, H. M. Osinga, E. J. Doedel, M. E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz and O. Junge. “A survey of methods for computing (un)stable manifolds of vector fields.” Int. J. Bifurcation and Chaos, Vol. 15(3), pp. 763-791, 2005.

[51] Y. A. Kuznetsov. “Elements of Applied Bifurcation Theory.” Second Edition, Springer Verlag, New York, 1998.

[52] M. G. Larson. “Notes on Triangulations and Refinements.” available by HTTP from http://www.phi.chalmers.se/education/courses/2000/detb-a-kf/Triang.ps, Chalmers University of Technology, Sweden, 2001.

[53] E. N. Lorenz. “Deterministic nonperiodic flow.” A method for estimating values that lie between two known values. Atmosph. Sc. Vol. 20, pp. 130-141, 1963.

[54] Maple: General-purpose commercial mathematics software package, developed by Waterloo Maple Inc., current version is Maple 11 released in February 2007.

[55] L. Mark. “Programming Python.” Second edition, O'Reilly & Associates, Inc., 2001.

[56] MATCONT, CL_MATCONT and CL_MATCONT_for_MAPS: continuation software in Matlab, developed under the supervision of W. Govaerts and Y. A. Kuznetsov, available by HTTP from http://www.matcont.ugent.be/ .

[57] MATLAB: A numerical computing environment and programming language, Created by “The MathWorks”, current version is MATLAB 7.4 released on March, 2007.

[58] J. R. Munkres. “Analysis on Manifolds.” Reading, MA: Addison Wesley, 1991.

[59] Oscill8: Dynamical Systems Toolset, v.1.12, developed by E. Conrad, available by HTTP from http://oscill8./-sourceforge.net/doc/quickstart.html , 2005.

[60] H. M. Osinga. “Non-orientable manifolds of periodic orbits.” In Proc. Int. Conf. Differential Equations, Equadiff 99(Berlin) Vol.2, eds. Fiedler B.,Gröger, K.& Sprekels, J. (World Scientific, Singapore), pp. 922-924, 2000.

[61] H. M. Osinga and B. Krauskopf. “Visualizing the structure of chaos in the Lorenz system.” Comput. Graph. 26, pp. 815-823, 2002.

[62] C. Perelló. "Intertwining invariant manifolds and Lorenz attractor." In Global theory of dynamical systems (Proc. Internat. Conf., Northwestern Univ., Evanston, Ill., 1979), Lecture Notes in Math. 819, Springer-Verlag, Berlin, pp. 375-378, 1979.

[63] Lorenz Equations in PlanetMath, available by HTTP from http://planetmath.org/-encyclopedia/LorenzEquation.html .

[64] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. "Numerical Recipes in C." Second edition, Cambridge University Press, 1995.

[65] S. S. Rao. "Applied Numerical Methods for Engineers and Scientists." Prentice Hall, 2002.

[66] W. C. Rheinboldt. "Numerical Analysis of Parametrized Nonlinear Equations." Wiley-Interscience, Lecture Notes in the Mathematical Sciences, University of Arkansas, 1986.

[67] A. H. Richard. "A First Course in Discrete Dynamical Systems.", 1996.

[68] R. D. Russell, and J. Christiansen. "Adaptive Mesh Selection Strategies for Solving Boundary Value Problems." SIAM J. Numer. Anal. Vol. 15, pp. 59-80, 1978.

[69] E. M. Schwarz and C. A. Krygowski. "The S/390 G5 floating point unit." IBM Journal of Research and Development, Vol. 43, No. 5/6, pp. 70-721, 1999.

[70] M. Segal and K. Akeley. "The OpenGL Graphics System: A Specification, Version 1.5." Silicon Graphics, Inc. 2003.

[71] R. Seydel. " Practical Bifurcation and Stability Analysis. From Equilibrium to Chaos."Second Edition, Springer Verlag, New York, 1994.

[72] D. Shreiner, M. Woo, J. Neider and T. Davis. "OpenGL Programming Guide." Second Edition, Addison-Wesley, 1997.

[73] C. Sparrow. "The Lorenz Equations: Bifurcations, Chaos and Strange Attractors." Appl. Math. Sci. No. 41., Springer-Verlag, New York, 1982.

[74] H. B. Stewart. "Visualization of the Lorenz system." Physica D: Nonlinear Phenomena, Vol 18, Issue 1-3, pp. 479-480, 1986.

[75] J. M. T. Thompson and H. B. Stewart. "Nonlinear Dynamics and Chaos." John Wiley, Chichester, New York, 1986.

[76] E. W. Weisstein. "MathWorld Resource." available by HTTP from http://math-world.wolfram.com/Manifold.html .

[77] H. Whitney. "Geometric Integration Theory." Princeton University Press, 1957.

[78] Wikipedia, the free encyclopedia, available by HTTP from http://en.wikipedia.org .

[79] P. Womack and J. Leech. "OpenGL Graphics with the X Window System, Version 1.3." Silicon Graphics, Inc. 1998.

[80] XPPAUT: ODE and bifurcation softwares, developed by G. Bard Ermentrout, available by HTTP from http://www./-math.pitt.edu/ bard/xpp/xpp.html .

[81] J. Yoshimi. "Dynamical Systems Theory." available by HTTP from http://www.-jeffyoshimi.net/research/researchDesc/DynamicalSystems.html , 2006.

[82] PLAUT04: Data visualization package for AUTO, developed by C. Zhang, Concordia University, 2004; see also [28, 83].

[83] C. Zhang. "Computation and Visualization of Periodic Orbits in the Circular Restricted Three-Body Problem." Master Thesis, Computer Science Department, Concordia University, 2004.

# Appendix A

# Implementation in AUTO of the Lorenz Manifold Calculations

## A.1 Implementation for the stable manifold of the origin

### A.1.1 Boundary value problem formulation

As explained in Section 6.2, the origin $\mathbf{0} = (0,0,0)^T$ is a saddle point with eigenvalues

$$\mu_1 \approx -2.66 \ , \quad \mu_2 \approx -22.8 \ , \quad \mu_3 \approx 11.82 \quad ,$$

and corresponding normalized eigenvectors

$$\mathbf{v}_1 \ , \quad \mathbf{v}_2 \ , \quad \mathbf{v}_3 \quad .$$

For computing the stable manifold of the origin, we seek solutions of the system $F(X) = 0$, where

$$F(X) \equiv \begin{cases} \mathbf{x}'(t) \ - \ T \ \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{x}(0) \ - \ d_0 \ ( \ \frac{\cos(\theta)}{|\mu_1|} ) \ \mathbf{v}_1 \ - \ \frac{\sin(\theta)}{|\mu_2|} \ \mathbf{v}_2 \ ) \\ T \ \int_0^1 \ || \ \mathbf{f}(\mathbf{x}) \ || \ ds \ - L \end{cases}$$

where $d_0$ is a small radius in the stable eigenspace centered at the origin. The starting vector $\mathbf{x}(0)$ is also controlled by the angle $\theta$, which is in turn controlled by a parameter $h$, namely, $\theta = 2 * \pi * h$.

Table A.1 summarizes the parameters used in the computation. The boundary conditions are listed in Table A.2.

| Parameter list | Representation |
|---|---|
| $PAR(1)$ | problem parameter $\rho$ |
| $PAR(2)$ | problem parameter $\beta$ |
| $PAR(3)$ | problem parameter $\sigma$ |
| $PAR(4)$ | radius $d_0$ |
| $PAR(5)$ | $h$, which controls the value of $\theta$, $\theta = 2\pi h$ |
| $PAR(6)$ | the first component of the starting point |
| $PAR(7)$ | the second component of the starting point |
| $PAR(8)$ | the third component of the starting point |
| $PAR(9)$ | $L_2$ *norm* $\equiv \|\|\mathbf{x}(1) - \mathbf{0}\|\| \equiv d_1$ |
| $PAR(11)$ | integration time $T$ |
| $PAR(12)$ | length $L$ |

Table A.1: Parameter list for the stable manifold of the origin

| No. | Boundary conditions |
|---|---|
| 1 | $BC(1) = x_1(0) - d_0\ (\cos(\theta) * (v_{1-1}) + \sin(\theta) * (v_{2-1}))$ |
| 2 | $BC(2) = x_2(0) - d_0\ (\cos(\theta) * (v_{1-2}) + \sin(\theta) * (v_{2-2}))$ |
| 3 | $BC(3) = x_3(0) - d_0\ (\cos(\theta) * (v_{1-3}) + \sin(\theta) * (v_{2-3}))$ |
| 4 | $BC(4) = x_1(1) - PAR(6)$ |
| 5 | $BC(5) = x_2(1) - PAR(7)$ |
| 6 | $BC(6) = x_3(1) - PAR(8)$ |
| 7 | $BC(7) = \|\|\mathbf{x}(1) - \mathbf{0}\|\| - PAR(9)$ |

Table A.2: Boundary conditions for computing the stable manifold of the origin

The number of boundary conditions is $n_b = 7$, there is one integral constraint, $n_q = 1$, and the dimension of the system is $n_d = 3$. The number of free parameters is

$$n_p = (n_b + n_q) - n_d + 1 = (7 + 1) - 3 + 1 = 6 \quad .$$

A detailed description of the steps in the computational procedure is given in Section 6.2.

## A.2 Implementation for the unstable manifold of the non-zero stationary point

### A.2.1 Boundary value problem formulation

The computational set-up for the unstable manifold of the non-zero stationary point is somewhat similar to that of the stable manifold of the origin. The two non-zero equilibria, denoted by $\mathbf{x}_{sp}^-$ and $\mathbf{x}_{sp}^+$ respectively, are approximately at

$$(x_1,\ x_2,\ x_3) \ = \ (\pm 8.485\ ,\ \pm 8.485\ ,\ 27) \quad .$$

Suppose again the corresponding normalized eigenvectors are

$$\mathbf{v}_1\ ,\quad \mathbf{v}_2\ ,\quad \mathbf{v}_3\quad ,$$

with $\mathbf{x}(0)$ near $\mathbf{x}_{sp}$, one of the non-zero stationary points, and $\mathbf{x}(0)$ is in the unstable eigenspace spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$.

We seek solutions of the system $F(X) = 0$, where

$$F(X) \ \equiv \ \begin{cases} \mathbf{x}'(t) \ - \ T\,\mathbf{f}(\ \mathbf{x}(t)\ ,\ \rho\ ) \\ <\mathbf{x}(0) \ - \ \mathbf{x}_{sp}\ ,\ \mathbf{w}_0 > \ - \tau_0 \\ <\mathbf{x}(1) \ - \ \mathbf{0}\ ,\ \mathbf{w}_1 > \ - \tau_1 \\ ||\ \mathbf{x}(0) \ - \ \mathbf{x}_{sp}\ || \ - \ d_0 \\ ||\ \mathbf{x}(1) \ - \ \mathbf{0}\ || \ - \ d_1 \\ T\ \int_0^1\ ||\ \mathbf{f}_{\mathbf{x}}(s)\ ||\ ds\ - L \\ f_3\ (\ \mathbf{x}(0)\ ,\ \rho\ ) \ - \ \psi_0 \end{cases} \qquad \text{(A.1)}$$

with, for example (see Run 4 in Section 6.3),

$$X \; = \; ( \, \mathbf{x}(\cdot) \; , \; d_0 \; , \; L \; , \; \tau_1 \; , \; d_1 \, ) \quad , \quad ( \text{ for fixed } \rho, \tau_0, \psi_0 \text{ and } T \, ) \quad .$$

A detailed description of the various steps in the computational procedure is given in Section 6.3.

Table A.3 summarizes all the parameters used in the computations while Table A.4 lists the boundary conditions.

| Parameter List | Name | Representation |
|---|---|---|
| $PAR(1)$ | $\rho$ | problem parameter |
| $PAR(2)$ | $\beta$ | problem parameter |
| $PAR(3)$ | $\sigma$ | problem parameter |
| $PAR(4)$ | $d_0$ | the distance from $\mathbf{x}(0)$ to $\mathbf{x}_{sp}$ |
| $PAR(5)$ | $d_1$ | the distance from $\mathbf{x}(1)$ to the origin |
| $PAR(6)$ | $\psi_0 : \; x_3'(0)$ | the derivative of the third component of $\mathbf{x}$ at time 0 |
| $PAR(9)$ | $\tau_0 : \; < \mathbf{w}_0, \mathbf{x}(0) >$ | the inner product of $w_0$ and $\mathbf{x}(0)$ |
| $PAR(9)$ | $\tau_1 : \; < \mathbf{w}_1, \mathbf{x}(1) >$ | the inner product of $w_1$ and $\mathbf{x}(1)$ |
| $PAR(11)$ | $T$ | integration time |
| $PAR(12)$ | $L$ | length |

Table A.3: Parameter list for the unstable manifold of the non-zero stationary point

| No. | Boundary conditions |
|---|---|
| 1 | $BC(1) = \; < \mathbf{x}(0), \; \mathbf{w}_0 >$ |
| 2 | $BC(2) = \; \|\mathbf{x}(0) - \mathbf{x}_{sp}\| \; - \; PAR(4)$ |
| 3 | $BC(3) = \; x_1(0) \, x_2(0) \; - \; PAR(2) \, x_3(0) \; - \; PAR(6)$ |
| 4 | $BC(4) = \; < \mathbf{x}(1) - \mathbf{0}, \; \mathbf{w}_1 > \; - \; PAR(9)$ |
| 5 | $BC(5) = \; \|\mathbf{x}(1) - \mathbf{0}\| \; - \; PAR(5)$ |

Table A.4: Boundary conditions for computing the unstable manifold of the non-zero stationary point

In this case, the dimension of the system $n_d$ is 3, the number of boundary conditions is $n_b = 5$ , and the number of integral conditions is $n_q = 1$. The number of free parameters $n_p$ is then

$$n_p = (n_b + n_q) - n_d + 1 = (5 + 1) - 3 + 1 = 4 \ .$$

## A.3 Format of output files

In AUTO, a summary of the computation is written in file fort.6, which usually corresponds to the window in which AUTO is run. The bifurcation diagram is contained in fort.7 and the complete graphics as well as restart data is contained in fort.8. Moreover, diagnostic messages, convergence history, eigenvalues and other informations are written in fort.9. AUTO also provides linux commands such as @sv, @ap, @df to manipulate these files. Corresponding commands also exist in the Python AUTO-interface. Usually one saves fort.7 as *b.∗*, fort.8 as *s.∗*, fort.9 as *d.∗*, where "*" denotes a user-selected name. The format of bifurcation file is somewhat self-explained, so we omit here. The format of solution files is briefly described below.

### A.3.1 Solution file

There are three parts of data in a solution file:

- data corresponding points in fort.7,
- the complete solution,
- the direction of the branch.

Only specified labeled solution points in a bifurcation file are written in complete data to the corresponding solution file. Specifically the following is written (see also Table A.5):

|  | Name | Representation |
|---|---|---|
| 1 | IBR | The index of the branch |
| 2 | NTOT | The index of the point on the branch |
| 3 | ITP | The type of the point |
| 4 | LAB | The label of the point |
| 5 | NFPR | The number of free parameters used in the computation |
| 6 | ISW | The value of ISW used in the computation(see [28]) |
| 7 | NTPL | The total number of points in the scaled time interval [0,1], NTPL = NCOL*NTST + 1 |
| 8 | NAR | The number of values written per point($NAR = NDIM + 1$) |
| 9 | NROWPR | The number of lines printed following the identifying line |
| 10 | NTST | The number of principal time intervals used in the discretization |
| 11 | NCOL | The number of collocation points used per principal mesh interval |
| 12 | NPARX | The dimension of the array PAR |

Table A.5: Information written in a solution file of BVPs

After the identifying line, there are $NTPL$ lines containing

$$\boxed{T\ ,\quad x_1(t_k)\ ,\quad x_2(t_k)\ ,\quad \dots\ ,\quad x_{NDIM}(t_k)\ ,\ \text{where } k = 1, 2,\ \cdots,\ \text{NTPL},}$$

where $x_1,\ x_2, \dots\ x_{NDIM}$ are the solution components.

Following this the indices and the values of the rate of change of the free parameters are printed:

$$\boxed{\begin{array}{l} ICP(I)\ ,\quad I = 1, \dots NFPR\ , \\ \dot{P}_i\ ,\quad i = 1, \dots NFPR\ , \end{array}}$$

This is followed by another $NTPL$ lines, each containing

$$\boxed{\dot{x}_1(t_k)\ , \quad \dot{x}_2(t_k)\ , \quad \ldots\ , \quad \dot{x}_{NDIM}(t_k)\ , \text{ where } k = 1, 2, \ \cdots, \text{NTPL},}$$

where $\dot{P}, \dot{x}$ denote the direction of the branch.

Finally the parameter values $PAR(i)$, $i = 1, ..., NPARX$ are written; see [28] for details.

Part of a solution file computed for the Lorenz system is shown below.

```
1   1   9   1   4   1   1201   4   2410   300   4   36
  0.0000000000E+00   8.2576232567E+00   8.5430449733E+00   2.6454467570E+01
  6.2286099431E-04   8.3151657399E+00   8.6424061259E+00   2.6466512206E+01
    ...                ...                ...                ...
  9.9182483371E-01   3.1037222949E-07   6.7829713844E-07   6.1899357610E-04
  1.0000000000E+00  -3.9056651480E-05  -8.5402962785E-05   3.2034410304E-04
1   4   11   12
  2.9624057355E-01  -1.1169042692E-01  -1.5108987550E-01  -2.7446210988E+01
  8.4996793955E-02   4.1546097711E-02   4.0095127497E-01
  7.6006384092E-02   2.6778012803E-02   3.9916258341E-01
    ...                ...                ...
  3.0534369579E-08  -2.6677454921E-07  -6.7398666011E-08
  2.8135642091E+01   2.6666666667E+00   1.0000000000E+01   7.2615639534E-01
  3.3382545204E-04  -1.1402161632E-11   0.0000000000E+00   0.0000000000E+00
  8.3245961545E-05   0.0000000000E+00   3.0077889420E+01   9.4697575186E+02
  0.0000000000E+00   0.0000000000E+00   0.0000000000E+00   0.0000000000E+00
  0.0000000000E+00   0.0000000000E+00   6.2831853072E+00   0.0000000000E+00
  0.0000000000E+00   0.0000000000E+00   0.0000000000E+00   0.0000000000E+00
  0.0000000000E+00   0.0000000000E+00   0.0000000000E+00   0.0000000000E+00
  0.0000000000E+00   0.0000000000E+00   0.0000000000E+00   0.0000000000E+00
  0.0000000000E+00   0.0000000000E+00   0.0000000000E+00   0.0000000000E+00
% Other data sets follow, if any
    ...                ...                ...
```

# Appendix B

# AUTO Utilities

## B.1 AUTO/Python integration

AUTO is a publicly available software for continuation and bifurcation problems in ordinary differential equations. The earliest version of AUTO was issued in 1980. From then on, it has evolved, with an X/Motif version AUTO94, a parallel version AUTO94P, AUTO97(with HOMCONT), and the C version AUTO2000. See [23, 35, 24, 25].

In order to make the package more powerful and more comprehensive, a new version AUTO-07p has been developed. This version is built on AUTO97, incorporating the Python interface, fixing some minor problems in the previous version, and adding some new features to enhance the utility of the package.

Python [55] is a popular object-oriented language used for both standalone programs and scripting applications in a variety of domains. It is free, portable, powerful, and remarkably easy to use.

The Python code is relatively self contained. The compiler is defined in `cmds/cmds.make`. This is basically a templated Makefile, used by both the old command language as well as by the Python code. In this section, we will explain how to combine AUTO97 and the Python interface.

### B.1.1 Filename conversion

In AUTO2000, all the filenames follow the 'easy to remember' convention, namely, all the bifurcation files start with *b.**, solution files start with *s.**, constant files start with *c.**, and homoclinic solution files start with *h.**. In order to be compatible with the Python interface, we use the same naming convention in AUTO-07p. Table B.1 lists the correspondences of the naming convention in AUTO97 (and previous versions) versus AUTO2000 and AUTO-07p.

| File type | AUTO97 | AUTO2000 |
|---|---|---|
| Constant file | r.* | c.* |
| Bifurcation file | p.* | b.* |
| Solution file | q.* | s.* |
| Homoclinic solution file | s.* | h.* |
| Diagnostic file | d.* | d.* |

Table B.1: Naming convention in AUTO97 versus AUTO2000 and AUTO-07p

### B.1.2 A utility program for filename conversion

Two simple utility Python scripts are provided for filename conversion. One is *rename.py*, which is used to simply rename the exist filename. Another is *replace.py*, which is used to replace each *p.** to *b.**, each *q.** to *s.**, each *s** to *h.** in relevant files. For use, one need to execute the following:

Correctly load the utility programs, which are located in:

**`AUTO_DIR/python/python_utilities/`**

Execute the script, with appropriate arguments. Below is an example:

```
//in the Makefile, change all the r. to c. then copy the file to Makefile1

>../replace.py r. c. Makefile Makefile1
```

For using *rename.py*, go directly to a folder under which a filename needs to be changed, and run the script. For example, go to AUTO_DIR/demos/ab/, change the constant filename *r.ab* to *c.ab* :

```
//change r.ab -> c.ab, r.ab.1 -> c.ab.1, etc.
>../rename.py
```

## B.2 Python CLUI and linux-based CLUI

AUTO-07p has two parallel CLUIs, namely, the newer python CLUI and the Linux-based commands in *AUTO_DIR/cmds*. One can use either CLUI, depending on one's preference.

## B.3 Python scripts for AUTO utilities

AUTO-07p also provide some files whose names end with *.auto*, which are Python scripts that can be run directly under the Linux CLUI. These python scripts automatically execute each consecutive run of the selected demo.

To use the Python CLUI for a new equation, change to an empty directory. For an existing equations-file, change to its directory. Do not activate the CLUI in the directory *AUTO_DIR* or in any of its subdirectories. Then type:

```
>auto
```

to activate the AUTO-07p CLUI. The screen will show a message which is similar to the following:

```
Python 2.4.3 (# 1, Oct 23 2006, 14:19:47)
[GCC 4.1.1 20060525 (Red Hat 4.1.1-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(AUTOInteractiveConsole)
AUTO>
```

Then one can type any Python command following the "*AUTO>*" prompt.

In the Linux CLUI, instead of running each step separately, one can type:

```
>auto lrz.auto
```

to automatically execute each run.

Below is a example, namely, *lrz.auto*, which computes two symmetric homoclinic orbits in the Lorenz equations.

```
# ===============
# AUTO Demo lrz
# ===============
print "\n***Compute stationary solutions***"
ld(e='lrz',c='lrz.1')
run()
sv('lrz')
print "\n***Compute periodic solutions***"
ld(c='lrz.2',s='lrz')
run()
ap('lrz')
print "\n***Compute the symmetric periodic solution branch***"
ld(c='lrz.3',s='lrz')
run()
ap('lrz')
print "\n***Clean the directory***"
cl()
```

The screen output will be of the following form:

```
Runner configured
f77 -c -O lrz.f -o lrz.o
```

```
f77 -O lrz.o -o lrz.exe /home/rchen/97/lib/*.o
Starting lrz ...
BR PT TY LAB   PAR(1)        L2-NORM      U(1)         U(2)         U(3)
1   1  EP  1  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
1   5  BP  2  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
1  13  EP  3  3.16000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
BR PT TY LAB   PAR(1)        L2-NORM      U(1)         U(2)         U(3)
2  42  HB  4  2.47368E+01  2.62685E+01  7.95602E+00  7.95602E+00  2.37368E+01
2  45  EP  5  3.26008E+01  3.41635E+01  9.17980E+00  9.17980E+00  3.16008E+01
BR PT TY LAB   PAR(1)        L2-NORM      U(1)         U(2)         U(3)
2  42  HB  6  2.47368E+01  2.62685E+01  -7.95602E+00 -7.95602E+00 2.37368E+01
2  45  EP  7  3.26008E+01  3.41635E+01  -9.17980E+00 -9.17980E+00 3.16008E+01
 Total Time    0.300E-01
lrz ... done
Saving fort.7 as b.lrz ... done
Saving fort.8 as s.lrz ... done
Saving fort.9 as d.lrz ... done
Starting lrz ...
BR PT TY LAB   PAR(1)      L2-NORM     MAXU(1)     MAXU(2)     MAXU(3)     PERIOD
4  20      8  1.48138E+01 1.23916E+01 1.17081E+01 1.39788E+01 2.27276E+01 1.62929E+00
4  35  EP  9  1.39266E+01 6.29547E-01 1.14802E+01 1.36631E+01 2.16345E+01 5.85401E+02
 Total Time    0.545E+00
lrz ... done
Appending fort.7 to b.lrz ... done
Appending fort.8 to s.lrz ... done
Appending fort.9 to d.lrz ... done
Starting lrz ...
BR PT TY LAB   PAR(1)      L2-NORM     MAXU(1)     MAXU(2)     MAXU(3)     PERIOD
6  20     10  1.48138E+01 1.23916E+01 -4.02246E-02 3.50485E-01 2.27276E+01 1.62929E+00
6  35  EP 11  1.39266E+01 6.29547E-01  8.84302E-12 3.26839E-01 2.16345E+01 5.85401E+02
 Total Time    0.535E+00
lrz ... done
Appending fort.7 to b.lrz ... done
Appending fort.8 to s.lrz ... done
Appending fort.9 to d.lrz ... done
```

## B.4 Activating the GUI in AUTO-07p

There is a Graphical User Interface in AUTO, developed by Xianjun Wang in 1994. It is not only a tool for creating or editing equations-files and constants-files, but it can also be used to run AUTO and to manipulate and plot output-files. Nevertheless, the GUI is not being used much recently. The reason is that, in its source code, in particular, its way to declare the variable-parameters list, is not supported by current C-compilers.

### B.4.1 Updating the source file

Under the folder *AUTO_DIR/gui/* , the file *auto97.c* gets updated. More precisely, the function *Wprintf()* has been modified. The updated version of *Wprintf()* and the previous version of *Wprintf()* are listed below:

Previous:

```
void Wprintf(va_list)
   va_dcl
   {
    Widget    w;
    va_list   args;
    char      *format,str[100],s[20];
    Arg       wargs[1];
    XmString  xmstr;
    int       i;
   }
```

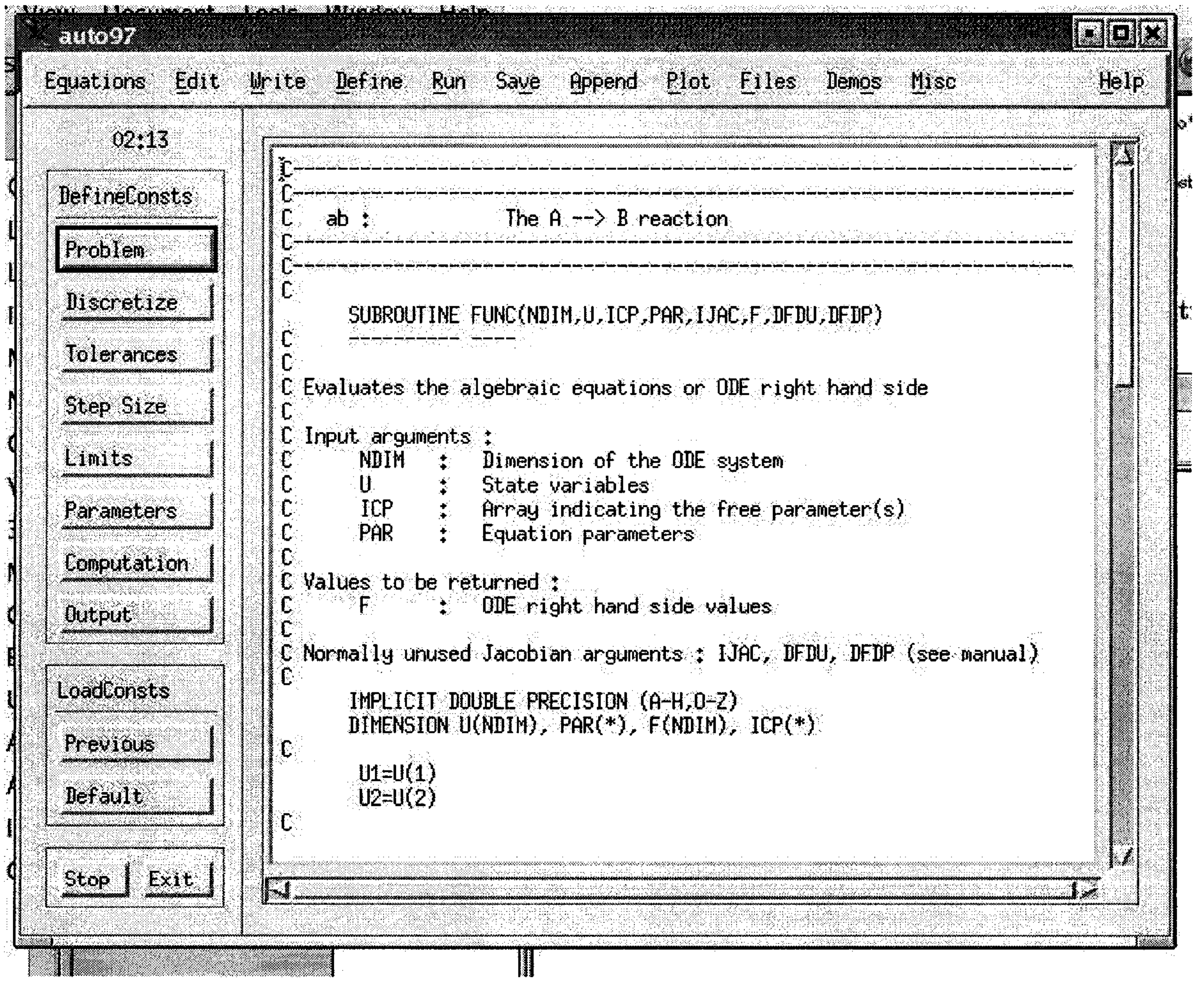
Current:

```
 void Wprintf(Widget w, ...)
  {
    va_list   args;
    char      *format,str[100],s[20];
    Arg       wargs[1];
    XmString  xmstr;
    int       i;
  }
```

After recompiling the source file, type

```
>@auto
```

The GUI appears as Figure B.1.



```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C   ab :            The A --> B reaction
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      SUBROUTINE FUNC(NDIM,U,ICP,PAR,IJAC,F,DFDU,DFDP)
C     ---------- ----
C
C Evaluates the algebraic equations or ODE right hand side
C
C Input arguments :
C      NDIM   :   Dimension of the ODE system
C      U      :   State variables
C      ICP    :   Array indicating the free parameter(s)
C      PAR    :   Equation parameters
C
C Values to be returned :
C      F      :   ODE right hand side values
C
C Normally unused Jacobian arguments : IJAC, DFDU, DFDP (see manual)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(NDIM), PAR(*), F(NDIM), ICP(*)
C
       U1=U(1)
       U2=U(2)
C
```

Figure B.1: GUI window

# Appendix C

# Packages related to AUTO

Packages that are related to AUTO include `CONTENT`, `DSTool`, `MATCONT/CL_MATCONT`, `XPPAUT`, *etc.* Below we give a brief introduction.

`CONTENT` [18] is designed to do simulation, continuation, and normal form analysis of dynamical systems. It supports bifurcation analysis of ODEs, iterated maps, and some evolution PDEs. Visualization of the solutions is possible in multiple 2D and 3D graphics windows.

`DsTool` [31] was created to advance the computation and visualization of dynamical systems. It can be used for both discrete and continuous systems.

`XPPAUT` [80] is a tool for solving differential equations, difference equations, delay equations, functional equations, boundary value problems, and stochastic equations. It incorporates the AUTO software.

`DDE-BIFTOOL` [19] is a Matlab package for the numerical bifurcation and stability analysis of delay differential equations with several fixed discrete and/or state-dependent delays. Periodic solutions, their Floquet multipliers and connecting orbits are computed as in AUTO, using piecewise polynomial collocation on adaptively refined meshes.

`MATCONT` [56] is a MATLAB package for the interactive numerical study of dynamical systems. It is developed in parallel with the command line continuation toolbox CL_MATCONT. Both MATCONT and CL_MATCONT allow to compute curves of equilibria, limit points, Hopf points, limit cycles and bifurcation points of limit cycles. The algorithms used are very similar to those used in AUTO; however AUTO is significantly faster.

A recent software called `Oscill8` [59] uses AUTO, CVODE, and other basic numerical software (LAPACK/BLAS/MINPACK). The current version runs on Windows, Linux or Mac. It has a graphical, interactive user interface and can add features and test very easily.