COMPUTER TECHNIQUES AND ALGORITHMS FOR DETECTION OF HETEROCLINIC CONNECTIONS IN EXPERIMENTAL DATA

by

Mark Grechanik, M.S.

THESIS

Presented to the Graduate Faculty of
The University of Texas at San Antonio
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Science and Engineering
Division of Computer Science
November 1998

COMPUTER TECHNIQUES AND ALGORITHMS FOR DETECTION OF HETEROCLINIC CONNECTIONS IN EXPERIMENTAL DATA

APPROVED BY SUPERVISING COMMITTEE:

Dr. Kay A. Robbins, Chair

Dr. Thomas C. Bylander

Dr. Steven Robbins

Dr. Mary Lou Zeeman

Acknowledgements

I first wish to thank the members of my committee, Dr. Kay A. Robbins, Dr. Steve Robbins, Dr. Thomas Bylander and Dr. Mary Lou Zeeman for their knowledge, encouragement and fruitful discussions. It has been a pleasure to work and learn from you. Special gratitude is reserved for Dr. Kay Robbins who tolerated me for almost five years and gently guided me through the forest of ideas and suggestions. Her help and patience made this thesis possible. My wife, Tina, has been a constant source of support, love, and friendship, making everything possible.

MARK GRECHANIK

The University of Texas at San Antonio

December 1, 1998

COMPUTER TECHNIQUES AND ALGORITHMS FOR DETECTION OF HETEROCLINIC CONNECTIONS IN EXPERIMENTAL DATA

Mark Grechanik, MS.

The University of Texas at San Antonio, 1998

Supervising Professor: Dr. Kay A. Robbins

This thesis presents computer algorithms and techniques for detection of heteroclinic connections in experimental data based on Stone–Holmes theory. Numerical experiments were initially conducted to determine the influence of box size and noise level prior to developing algorithms to detect heteroclinic connections in real physical systems. The real system studied in this thesis is a flat hydrocarbon–air flame on a porous plug burner. The flame front organizes in spatio–temporal patterns, and the parameter space in which these patterns occur is immense. Intermittent ordered patterns in the flame systems are identified as heteroclinic connections. This thesis describes algorithms and techniques and the software that implements them based on neural computation algorithms. After finding characteristic parameters of the dynamic system, differential energy is computed. Digitization techniques are applied in order to transform analog differential energy data into a digital array. Finally, a neural network is used to recognize exponentially growing and declining patterns at the beginning and the end of the suggested heteroclinic pattern. Later additional verification techniques are used to determine whether the detected exponential pattern is truly heteroclinic. The approach is applied to a specific system, however, there are many systems that can be transformed to those having similar basic behavior and functionality. Some experimental results are given that show good correspondance with theoretical predictions.

Contents

Chapter 1

Introduction

1.1 General Description of the Problem

This thesis introduces computer techniques for detection and analysis of heteroclinic connections and intermittency in experimental data. The terms *heteroclinic* and *intermittency* arise in the theory of dynamical systems. In this thesis they are the underlying concepts upon which the theoretical research and practical computing techniques are built. The techniques allow the investigation of behavior in complex dynamical systems where standard analytical methods are difficult to apply.

Although theorems later in the thesis give strict and explicit definitions, it is important to have a sense of their meaning to understand the nature of the work described in the thesis. *Webster's New World Dictionary And Thesaurus* defines "intermittency" as "stopping and starting again at intervals; periodic". The same reference gives the definition of "chaos" as "extreme confusion or disorder". In the context of dynamical systems, intermittency usually implies nonperiodic or nondeterministic behavior interrupted with periods of regular behavior. Many types of intermittency have been observed in physical systems some of which are described in Section 1.2. This thesis focuses on a particular type of intermittency that is associated with hetero-

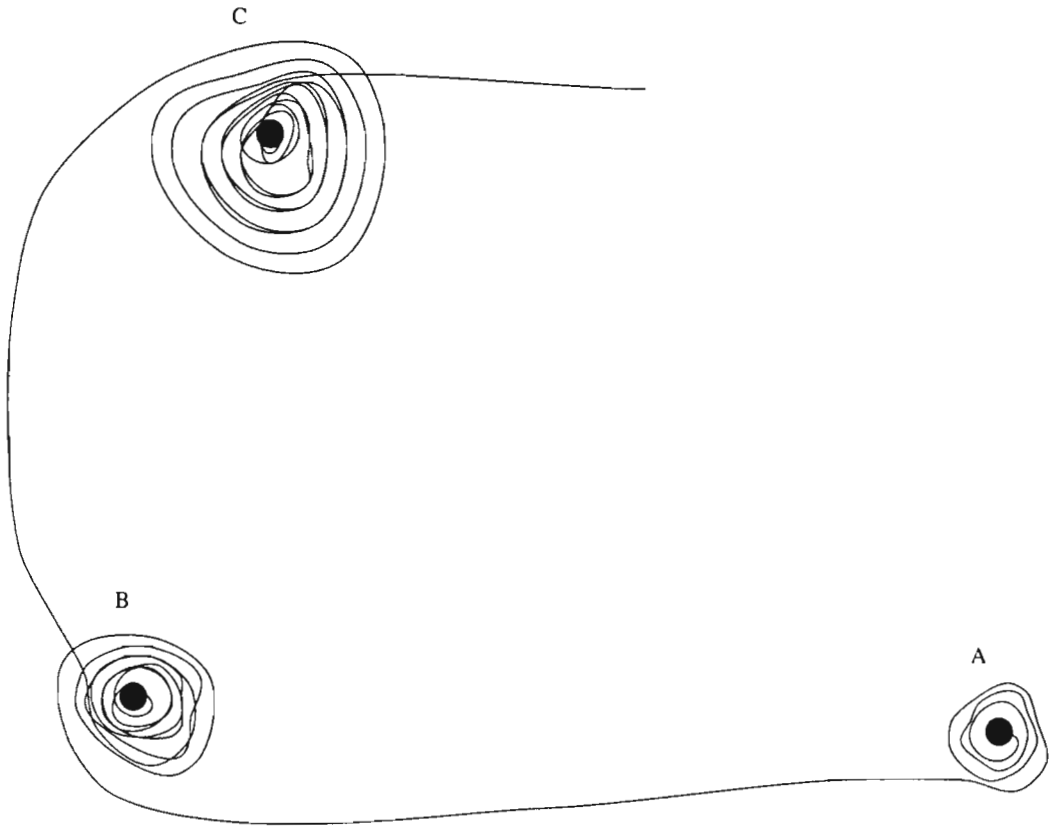

Figure 1.1: Type of behavior observed in a system having heteroclinic connections between a series of fixed points A, B, and C.

clinic connections found in dynamical systems [1]. Consider a dynamical system described by system variables. The values of these variables at each moment of time is called the state of the system. The graph of states of a dynamical system with heteroclinic connections shown in Figure 1.1. The graph shows that the system stays for a while near state A and then rapidly moves to state B. The system stays there again for a long interval of time, and afterwards moves to some new state. Eventually, the system may visit one of the previous states. The system showing such behavior is said to have heteroclinic connections. The existence of heteroclinic connections is difficult to prove even for simple dynamical systems. Techniques such as the Melnikov method

[2] are beyond the scope of the thesis. Heteroclinic connections are generally unstable and exist in a narrow region of parameter space. In computational models it is difficult to find them because of the finite precision of computation. The finite precision of the experiment affects the existence of heteroclinic connections the same way as the finite precision of computation does. For a nonlinear system a small additive noise may influence its behavior at a long time interval, and this is exactly what happens with heteroclinic connections. Thus, an important question is how to stabilize the heteroclinic connections in a dynamic system and extract information about them.

A first step in this direction was provided in Stone et al [3, 4] where evidence of heteroclinic connections was found in dynamical systems in the presence of symmetry. It was shown that the requirement that a vector field and perturbations of it are symmetrical, implies that heteroclinic connections can occur in a structurally stable manner. Several studies of various dynamical systems showed the existence of heteroclinic connections [5, 6, 7]. Stone's work [1] is the starting point of this thesis and is discussed in Chapter 2.

In retrospect, it is clear that scientists and engineers knew about chaos in dynamical systems for a long time. They called it "noise" or "turbulence". The so called "factor of reliability" was used to determine whether a system's behavior was deterministic or not. During the last four decades significant steps have been made in understanding the nature of dynamical systems. The availability of powerful computers allowed researchers to obtain results reflecting the complex nature of even simple dynamical systems. In order to analyze a complex physical system like the one presented in this thesis, it should be modeled first. The art of modeling a complex physical system by a less complex physical system described by a known set of differential equations is a well established practice. Depending on how well the factors were considered, the resulting model system may correspond to the original system with some degree of accuracy. The model system may predict new results or validate known results. Even though modelling by differential equations is thought to be viable, the approach has a serious glitch. A system can be mathematically deterministic even though it is not experimentally deterministic.

The mathematical determinism means that for a specific initial condition both the system's and model's solutions satisfy the descriptive set of differential equations. Experimental determinism fails when it is impossible to measure initial conditions with infinite precision. Thus, the farther a system develops in the future the bigger the difference between the mathematically determined solution characterizing the system's state and the experimentally determined system state. It should be noted that the lack of predictability is not a fault of the model, but an expression of a real physical fact of life. Thus, it may be worth studying how to find some determinism in the unpredictable aspects of the system's behavior that may become predictable under certain conditions.

1.2 Previous Research in Intermittency

Homoclinic and heteroclinic orbits are important in predicting the global behavior of dynamic systems. Generally speaking, a review of previous research must start with the work of Henri Poincaré. It was his genius that supplied many of the current methods for exploring the unexpected wonders of even "simple" (low–order) dynamic systems. Poincaré emphasized the importance of obtaining a global, qualitative understanding of the character of a system's dynamics. One of the basic concepts used throughout this thesis is the Poincaré map. It can be understood by considering numerical integration of the equations of motion using a known method of numerical integration such as a fourth order Runge–Kutta method. The integration requires an initial value of the state x_0. After the equation is integrated with one step Δx, it will yield $x_1 = x_0 + \Delta x$, and hence the map $x_0 \rightarrow x_1$. Thus, the solution described by a continuous flow is transformed into a *discrete map* generally written as $x_{n+1} = g(x_n) = g(g(x_{n-1}))$, where g is some function. This map is called a Poincaré map.

One of the first papers published about heteroclinic connections and intermittency was [28]. May [8], and Manneville, and Pomeau [9] described an intermittent form of chaos that is reminiscent of the intermittency found in one–dimensional maps. Manneville, and Pomeau

[9] used the Lorenz system as a mathematical model. This is a system of three nonlinear ODEs

$$\begin{cases} \dot{x} = & \sigma(y-x) \\ \dot{y} = & -xz + rx - y \\ \dot{z} = & xy - bz \end{cases} \tag{1.1}$$

where r, σ, and b are three parameters; r is related to heating convection, σ is the Prandtl number and b is related to wave number of a hydrodynamic system. The reduced Rayleigh number r is a control parameter. The Lorenz system was obtained by simplification of a more complex system governing finite amplitude convention in a fluid layer heated from below. The Lorenz system showed very complex behavior, despite the seeming simplicity of the mathematical description. In particular, detailed research was carried out for $\sigma = 10$, $b = \frac{8}{3}$ and $145 < r < 170$. It showed that for $r = 145$ a typical aperiodic orbit exists which is being transformed while increasing $r \rightarrow 149$. The destabilization of the orbit happens through intermediate "turbulent" bursts. In order to analyze this phenomenon a Poincaré map was used. It can be thought of as a hypersurface. Each time a trajectory intersects with the surface, the place is marked with a dot. So this procedure is similar to the one producing a stroboscopic vision of a motion. So far it makes sense to apply an iterative map procedure $y_{n+1} = f(y_n)$ in order to find transversal relationships among the points. It is done by choosing the plane $(x = 0)$ and defining the section of interest by setting $(z = z_{const})$. The results are shown in Figure 1.2. The evolution of the graph when changing r is shown in Figure 1.3. The Poincaré map for $r = 166.2$ is presented in Figure 1.2. This is mainly due to the fact that for $r = 166.2$ the system explores a much larger region of phase space and jumps from one part of the hyperspace to the others. The region of the map outlined with a solid line rectangle in Figure 1.2 is shown enlarged in Figure 1.4. The Figure 1.4 shows a parabolic smooth curve which is the direct descendant of the parabolic curve shown in Figure 1.2. So when the system's state is located close enough to the bisectrix it will take a long time for the system to move away from the bisectrix neighborhood. For an external observer the system would seem to be frozen. However, the changes in the system's behavior is so small that when

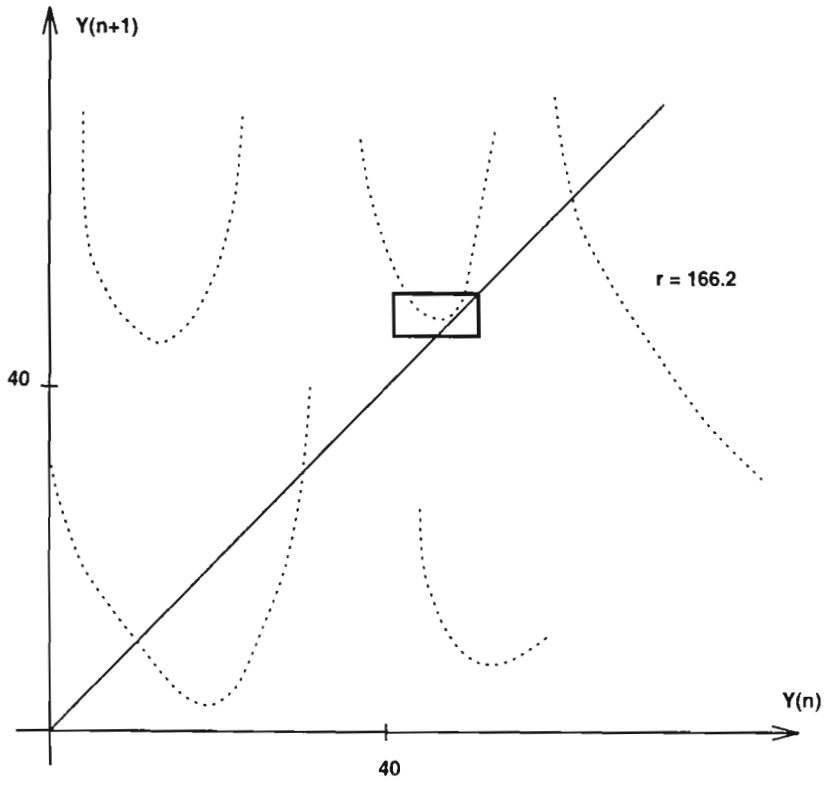

Figure 1.2: One–dimensional map applied to Lorenz system.

the system finally gets out of the bisectrix neighborhood on the graph where the state function is heavy nonlinear, the behavior will seem to change rapidly. It may take a long time depending on the state function before the system will be able to return to the initial behavior. When the curve becomes tangent to the bisectix a "laminar" flow is observed. The system is said to have a homoclinic connection in this case. In order to extract numerical information from a physical system some measuring tools must be applied. In this particular case the task can be described as a motion estimation, measurement, and analysis one.

Numerical systems like the Lorenz system have very few state variables in most cases. They are relatively easy to simulate and analyze. It is an opposite thing when a real physical dynamic system is considered. Such systems have many state variables, making it difficult and

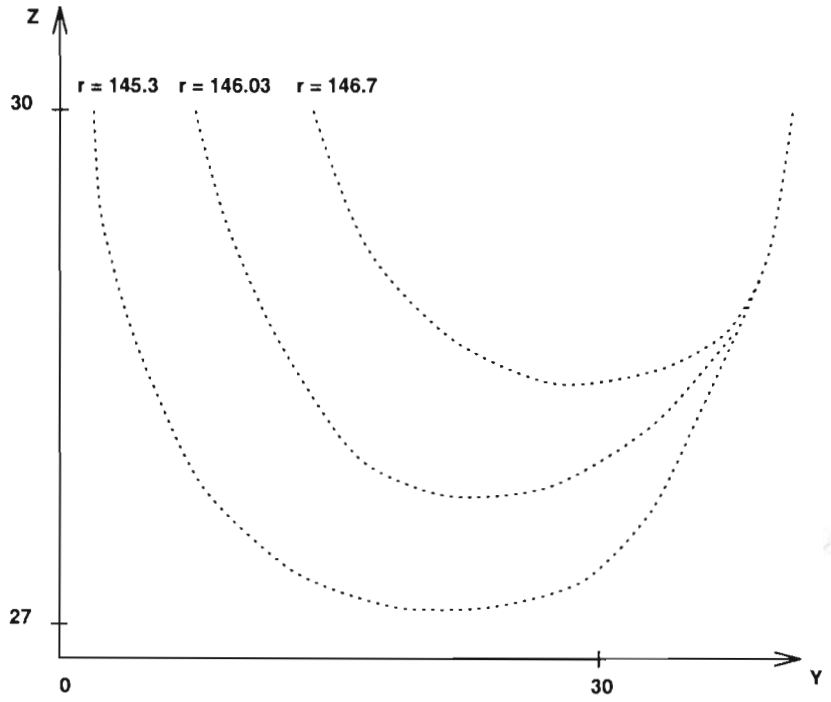

Figure 1.3: Evolution of the graph when changing r.

interesting at the same time to investigate their behavior.

1.3 Overview of the Thesis

The system studied in this thesis is a flat hydrocarbon - air flame in the output of a porous plug burner. Different profiles of burner have been used and three principal parameters are varied - fuel, flow rate, and equivalence ratio. The parameter space in which these dynamic modes occur is immense. Some of the modes are intermittent, the system demonstrates rapid changes in behavior and its orbit has heteroclinic connections. Flame images are captured by a video camera and saved on a videotape. The videotape is digitized later and converted into a number of binary files saved on a hard drive of a workstation. The research conducted in this thesis is based on the same ideas that are described in Stone et al [10]. In this paper intermittent

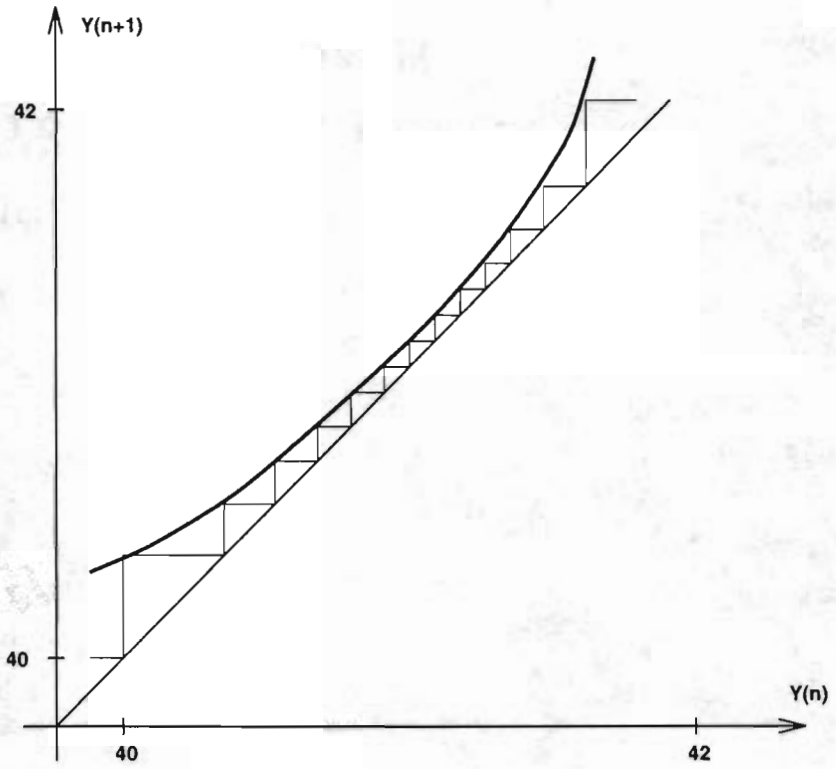

Figure 1.4: One–dimensional tangent bifurcation structure.

ordered patterns in flame systems were identified as heteroclinic connections. An example of ordered patterns in a flame system is shown in Figure 1.5. It introduces a technique for detecting heteroclinic connections in experimental data based on theoretical works on passage time distributions. Stone et al present a discussion of the characteristics of pattern–forming systems exhibiting heteroclinic connections, emphasizing the importance of the passage time of the intermittent pattern and its relationship to the physics of unstable equilibria. The skewed distribution of passage times with exponential tail is a distinctive signature of systems in which a trajectory is repeatedly injected near the group of stable orbits in the presence of noise.

This thesis addresses how intermittent states can be tracked and analyzed from videotape by applying image processing and computer animation techniques. The goal of this work is to confirm the association between heteroclinic connections and intermittency and to develop

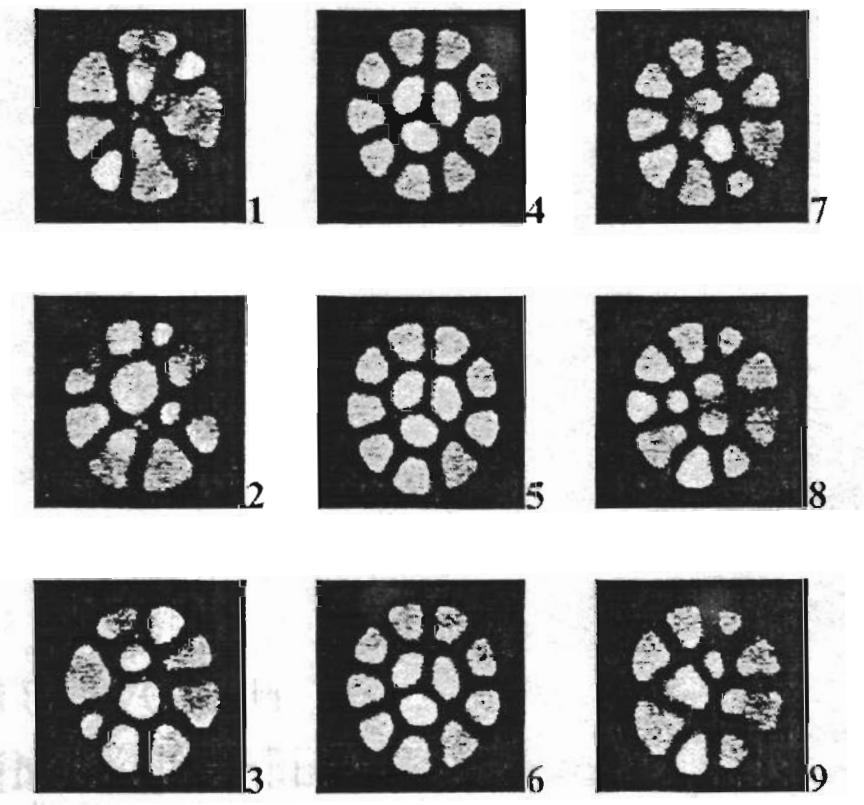

Figure 1.5: Sample frames of ordered spatio–temporal flame patterns.

practical tools for measuring heteroclinic connections and intermittency in experimental data. This work is a part of an extensive research project to analyze the combustion experiment. The results of this thesis will allow these ongoing projects to expand their potential database and be able to provide more extensive findings in an automated manner.

The thesis is organized as follows. Chapter 1 gives a general introduction of the problem and discusses the previous research. Chapter 2 provides theoretical background on heteroclinic connections and intermittency and the application of Stone - Holmes theory to the Fokker - Planck equation. Chapter 3 describes numerical experiments that were performed to understand the parameters that must be set when applying Stone–Holmes theory to real systems. Chapter 4 describes the proposed computer techniques for detection and analysis of heteroclinic connections in sequences of flame images. The design and necessary structures needed to visualize intermittent behavior are explained in this chapter. The main purpose of Chapter 4 is to show how to analyze intermittency in spatial temporal data. In chapter 5 experiments are carried out using the methods and algorithms which are described in Chapter 4 to determine time borders of intermittent behavior. Chapter 6 provides conclusions and possible extensions to the work started by this thesis.

Chapter 2

Theoretical Background

2.1 Introduction

Much of the early research in nonlinear systems focused on nonlinear oscillations described by second order ODEs. Their behavior can be described by a phase plane that is very informative. Jackson [11] points out five major steps that can be followed when analyzing second order nonlinear equations:

- Locate all of the *fixed points* of the dynamics (the equilibrium points). These are also referred to as the *singular points*.
- Determine the character of the flow in the neighborhood of each fixed point. This involves a simple linear analysis about each fixed point.
- Use the equation $\frac{dx}{dy} = \frac{P(x,y)}{Q(x,y)}$, which gives the tangent of the orbit at each point, to obtain the global topological character of the flow. This step connects the local flows found in the previous step.
- In some cases there may be a need to determine an integrating factor to explicitly integrate $\frac{dx}{dy} = \frac{P(x,y)}{Q(x,y)}$. This integration gives a function $K(x,y) = K_0(x_0, y_0)$ that determines the

orbit through each point (x_0, y_0) whose topological character is already known. $K(x, y)$ is the only time–independent integral of motion of the system.

- The final piece of information is the time dependence along these orbits, which may make use of $K(x, y)$ to eliminate one variable from one of equations of the system. In most cases this detailed information can be obtained only by numerical integration of these equations.

2.2 Linearization around a fixed point

Study of higher dimensional system begins with the study of system fixed points. Suppose that a nonlinear system is described by an ordinary differential equation of the form

$$\dot{x} = f(x) \tag{2.1}$$

where $x \in \Re^n$. The fixed points or steady solutions satisfy $f(\overline{x}) = 0$. In order to illustrate the notion of a "fixed" or "stable" point consider the phase portraits of the two–dimensional system shown in Figure 2.1, Figure 2.2, and Figure 2.3. Representative system orbits are shown with arrows. In the case of a stable fixed point, all trajectories tend to move toward the point. In the case of an unstable fixed point, all trajectories move away from the point. In the case of a saddle point in a two–dimensional system, only four orbits connect with a fixed point. An example of a dynamic system with a saddle point is described by following system of ODEs $\dot{x} = x, \dot{y} = -y$. An orbit which converges to the saddle point as $t \to +\infty$ is called an *incoming separatrix*, whereas if it converges to the saddle point as $t \to -\infty$, it is called an *outgoing separatrix*.

In the case of a saddle point it is very important to find out how a nonlinear system behaves in the neighborhood of the fixed point. The established procedure for determining the behavior of the nonlinear system is to examine the linearized system. The procedure for linearizing 2.1 at $\overline{x}$ is to substitute $x = \overline{x} + \xi$ into the equation, expand in a Taylor series, and

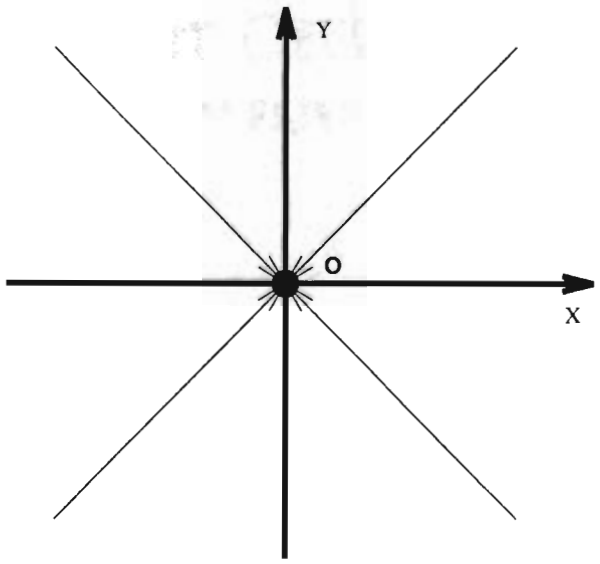

Figure 2.1: Phase plane of a system with a stable fixed point.

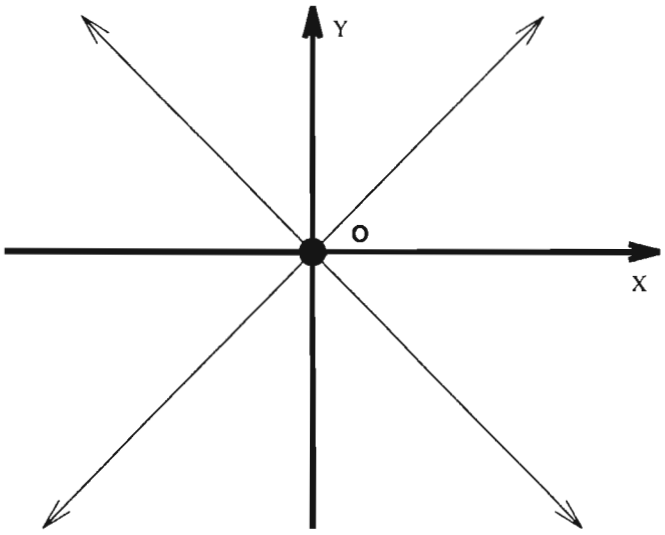

Figure 2.2: Phase plane of a system with an unstable fixed point.

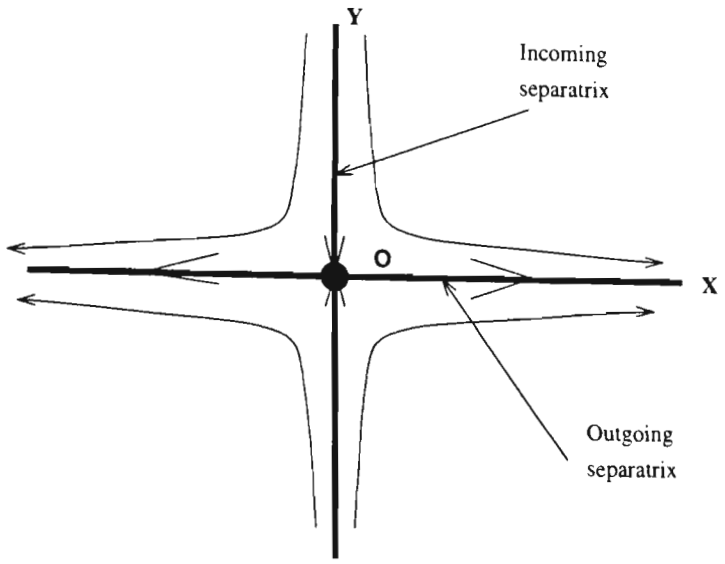

Figure 2.3: Phase plane of a system with a saddle point.

throw away terms that are nonlinear. The resulting system is

$$\dot{\xi} = Df(\overline{x})\xi, \qquad \xi \in \Re^n \tag{2.2}$$

where $Df = [\frac{df_i}{dx_j}]$ is the Jacobian matrix of the first partial derivatives of the function $f = (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n))$. Since 2.2 is a linear system, the solution can be easily obtained by integration. The first integral of 2.2 gives the the linearized flow map $D\phi_t(\overline{x})\xi$ at a fixed point $\overline{x}$.

2.3 Manifolds

The dynamics of an n–dimensional nonlinear system in the phase space may be restricted to *manifolds* that are embedded in R^n. Any k–dimensional region continuously embedded in n–dimensional Euclidean space defined by equations $M_j(x_1, \ldots, x_n), (j = 1, \ldots, n - k) = 0$ is called a "manifold". If the functions $M_j(x)$ are differentiable, then the manifold is

called a differentiable manifold. Stable and unstable manifolds are used to describe the system behavior near a chosen point p. In this thesis the basic system is written as

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

where $x = x(t) \in R^n$ is a vector function of the independent variable t and $f : U \rightarrow R^n$ is a smooth function defined on some subset $U \subseteq R^n$. f is a vector field that generates a flow $\phi_t : U \rightarrow R^n$, where $\phi_t = \phi(x,t)$ is a smooth function defined for all x in U and t. Now a definition for the stable and unstable manifolds can be given. Let's denote a fixed point p and correspondingly the stable and unstable manifolds of this point as $W^s(p)$ and $W^u(p)$. The set of all points that constitute a flow $\phi_t(x)$ converging to point p is called the stable manifold of p. Correspondingly, the set of all points that constitute a flow $\phi_t(x)$ diverging from point p is called the unstable manifold of p. It can be written as $W^s(p) = \{x \in U | \phi_t(x) \rightarrow p \text{ as } t \rightarrow \infty,$ and $\phi_t(x) \in U$ for all $t \geq 0\}$, $W^u(p) = \{x \in U | \phi_t(x) \rightarrow p \text{ as } t \rightarrow -\infty \}$, and $\phi_t(x) \in U$ for all $t \leq 0\}$.

As an illustration of the ideas of the stable and unstable manifolds, let's take a look at the equation $\ddot{x} + \delta\dot{x} - \beta x^2\dot{x} - x + x^3 = 0$ that will be used later in the thesis. When $\delta = 0$ then $W^s(p)$ and $W^u(p)$ are saddle separatrices. When $\delta > 0$ then the solution of this equation consists of transverse homoclinic orbits as long as the manifolds intersect. An example of solution space divided by manifolds for this equation is shown in Figure 2.4. A true heteroclinic dynamic system is not structurally stable. However, if a dynamic system is symmetric, then its heteroclinic cycles may be stable. A heteroclinic trajectory is a trajectory that is located on the intersection of the stable and unstable manifolds.

2.4 Wiener Process

In order to apply Stone–Holmes theory to a numerical dynamic system, its trajectory should be perturbed by small fluctuative noise. A random process with a small amplitude is

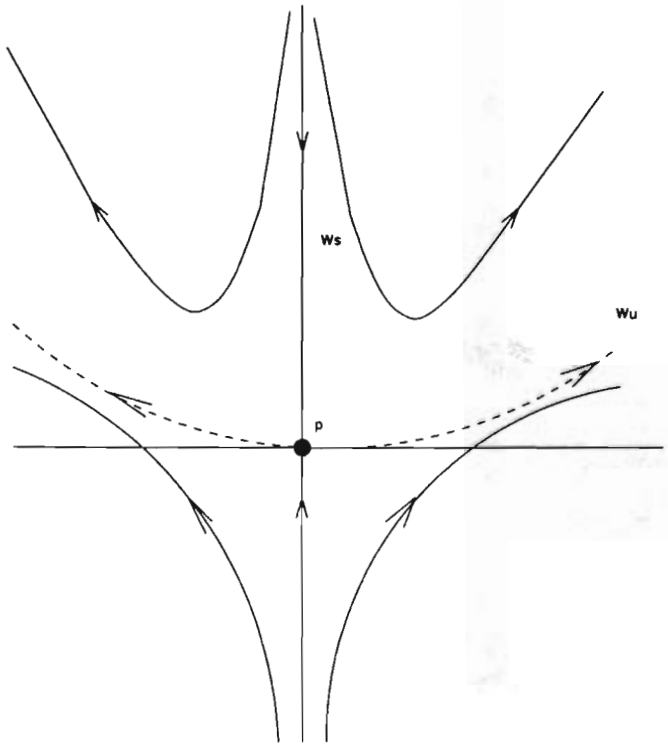

Figure 2.4: Example of a phase space divided by the stable and unstable manifolds of the fixed point p.

applied to a trajectory of the system so that the trajectory is forced to make small fluctuations around the border. The process is called Wiener process. The mathematical description of the process was given by N.Wiener in the 1920's. The physical root of this process lies in a discovery made by the English botanist R.Brown, when he analyzed the random motion of a small colloidal-size particle submerged in water.

Before the strict mathematical definition of Wiener process is given, let's define some properties of the randomly moving particle starting at the point $x = 0$ at time $t = 0$. The movements occur only in one-dimensional space. Denote the point's position on the line at time $t > 0$ as $\xi(t)$. Without loss of generality it can be assumed that $\langle \xi(t) \rangle = 0$ for all $t \geq 0$. If the temperature of the water remains constant, the distribution of any increment $\xi(t+s) - \xi(t)$ should not depend on t. The definition of a Wiener process is the following [12]:

Definition 2.1 *A standard Brownian motion or Wiener process is a stochastic process* $\{\xi(t); t \geq 0\}$ *having the following properties:*

- $\xi(0) = 0$;
- $\xi(t); t \geq 0$ *has independent increments;*
- *for any* $0 \leq s < t$,

$$P\{\xi(t) - \xi(s) \leq x\} = \frac{1}{\sqrt{2\pi(t-s)}} \int_{-\infty}^{x} e^{(-\frac{u^2}{2(t-s)})} du \tag{2.4}$$

Later in the thesis Wiener processes are used to make the equation being considered here stochastic in order to keep system trajectories in close proximity to the ideal heteroclinic orbit.

2.5 Ornstein–Uhlenbeck Process

As it was mentioned above, a rigorous mathematical theory of Brownian motion was developed by Wiener. A drawback of this theory is that physical details were mostly discarded. Subsequently the theory could not be applied to a real physical system to calculate a path of a single particle. To eliminate this drawback, Ornstein and Uhlenbeck developed a model similar to the Wiener process which was closer to the physical world. In this section a description of the process is given and the connection to the numerical experiment model is traced.

Consider a particle in liquid at some instance of time under Brownian motion. The mass of the particle is m, and it is moving with velocity $v(t)$ at time t. Two forces affect the particle. The first force is the frictional force of the liquid. According to Stoke's law it is proportional to $-v(t)$ and is given by $-\beta v(t)$ where $\beta > 0$ is a constant that depends on such factors as the viscosity of the liquid and the diameter and mass of the particle. The second force affecting the particle's movements is due to the effect of the molecular bombardment. It results in instantaneous random changes in the acceleration of the particle. This force can be represented by a Wiener stochastic process denoted by $w(t)$. According to Newton's second law the equation of motion for the particle can be written:

$$m\Delta v(t) = -\beta v(t)\Delta t + \Delta w(t) \tag{2.5}$$

Let's assume that $w(0) = 0$ and the following conditions hold:

- the stochastic process $w(t), t \geq 0$ has independent increments;
- the distribution of $w(t+s) - w(t)$ depends only on s;
- the sample paths of $w(t)$ are continuous with the probability 1.

Since $w(t)$ is Wiener process and assuming that $\langle w(t) \rangle \equiv 0$ and putting $\langle w(t)^2 \rangle = \sigma^2 t$ it is possible to write

$$w(t) = \sigma\xi(t), \tag{2.6}$$

where $\xi(t)$ is standard Brownian motion. Substituting 2.6 into 2.5, dividing by Δt and letting $\Delta t \to 0$ the following equation can be derived

$$m\frac{dv(t)}{dt} = -\beta v(t) + \sigma\frac{d\xi(t)}{dt}. \tag{2.7}$$

In [12] the solution of 2.7 is given along with the following definition of Ornstein–Uhlenbeck process.

Definition 2.2 *The stochastic process $v(t); t \geq 0$, where $v(t)$ is given by*

$$v(t) = \frac{\sigma}{m}(v_0 e^{-\beta t} + \xi(t) - \beta e^{-\beta t}\int_0^t e^{\beta s}\xi(s)ds) \tag{2.8}$$

is called the Ornstein–Uhlenbeck process.

After integrating 2.8 by parts, the average velocity of a Brownian particle can be written as $\langle v(t)\rangle = \frac{\sigma}{m}v_0 e^{-\beta t}$. In other words, the Ornstein–Uhlenbeck process expressed by equation 2.8 is a process described by a linear ordinary differential equation with an additional stochastic process, which is an integral part of this system. Having applied the definition of Ornstein–Uhlenbeck process to the system describing the linear part of the heteroclinic trajectory with additive Wiener noise, the system can be considered as an Ornstein– Uhlenbeck process with some particular properties unique for the process. The analytical analysis of the probability distribution depending on a distribution law of an initial condition is done using the Ornstein–Uhlenbeck model.

2.6 The Results of Stone-Holmes Theory

We now give a definition of homoclinic and heteroclinic orbits. A point q is called homoclinic to a fixed point p if the orbit $\varphi_t(q)$ based at q approaches p as $t \to \pm\infty$. A point q is called heteroclinic to a pair of points p^+, p^- if the orbit $\varphi_t(q)$ based at q approaches p^+ as $t \to +\infty$ and p^- as $t \to -\infty$. One of the reasons that real–life systems with homoclinic

(heteroclinic) connections are not well studied is indicated by the definitions. A system must be observed for a long period of time. In the meantime even a small disturbance may kick the system off the homoclinic (heteroclinic) cycle.

A symmetric system with heteroclinic cycles is picked for this analysis because the presence of the symmetry makes the heteroclinic cycles structurally stable. Stone and Holmes [1] estimates were derived for the mean recurrence time of orbits in the neighborhood of an attracting heteroclinic orbit in an ordinary differential equation. The equation is a subject to an additive small random noise. Stone [4] obtained the probability distribution of times during which a homoclinic or heteroclinic orbit stays in a small box surrounding a saddle point as a function of the unstable eigenvalue at the saddle. Measurements of this distribution in experimental data provide an estimate of the unstable eigenvalue and help to give insight into the stability of the system as explained in this section.

Aubry et al [3] assumed that the behavior of an orbit near a heteroclinic connection is dominated by the system's behavior at the hyperbolic saddle points. Thus, the analysis was reduced to a linear process. The center point of the theory is the box experiment. In Figure 2.5, saddle point O is surrounded by a small area box. A trajectory enters the box, passes through it, and leaves the box. The whole passage takes some time that varies for each entry of the trajectory into the box. The time that the trajectory spends in the box is called **passage time**. It was noticed that a law exists governing the passage time distribution when the system has a heteroclinic trajectory. Such systems happen to be intermittent. Intermittency can be described as a system with trajectories that reach some stability, stay for a while in this state, and then rapidly changes their behavior.

After assumptions were made about a random process and topological properties of eigenvalues, the standard **Fokker-Planck** equation for the linear process resulted. This equation has a well-defined solution obtained in Breiman [13]. It describes the conditional probability density function for solutions of ODE of the process. Based on initial and boundary conditions

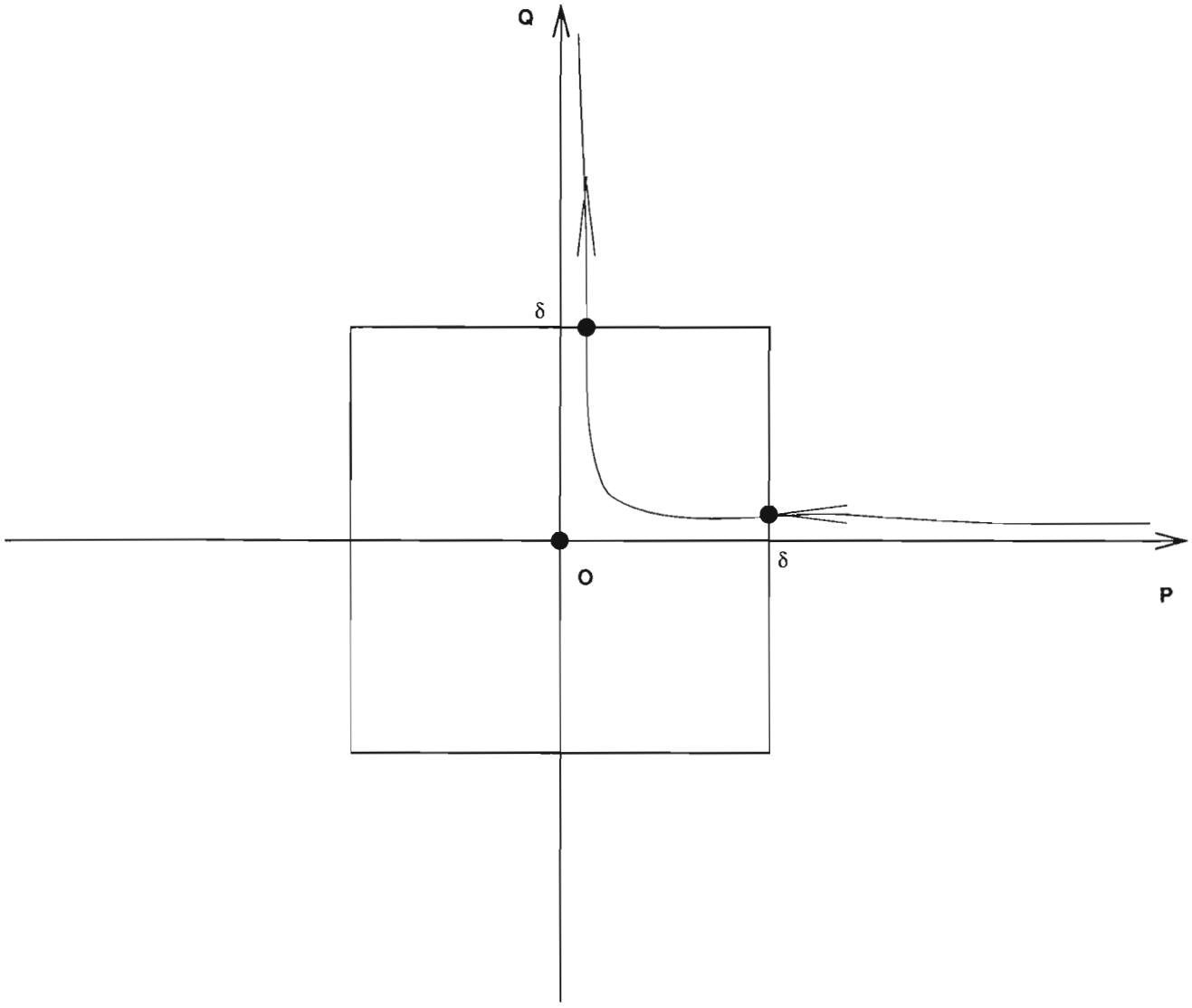

Figure 2.5: Illustration of the box experiment.

he derived a formula for the probability distribution for passage times

$$P(t) = \frac{2\lambda_u \Delta(t) e^{\Delta_2(t)}}{\sqrt{\pi}(1 - e^{-2\lambda_u t})} \tag{2.9}$$

where

$$\Delta(t) = \delta(\frac{\varepsilon^2}{\lambda_u}(e^{2\lambda_u t} - 1))^{-\frac{1}{2}},$$

δ is a half-size of a side of the box, ε is the noise amplitude, and λ_u is an unstable eigenvalue. The papers [3, 4] include several numerical simulations of a linear system and Duffing's equation. Both systems were subject to a random noise process. There were several dependencies obtained as a result of the simulation. The main characteristic dependency was a dependency of mean passage time on the level of the noise. Numerically generated graphs for conditional probabilities were built showing a fair coincidence with the graph built using 2.9.

2.7 Passage Time Theory Without Noise

As mentioned in the previous section, the analysis of a system with a heteroclinic orbit near a critical point may be reduced to the analysis of a linear process. In this section an equation is derived for estimating the distribution of the passage time probabilities in a system with heteroclinic connections without noise. Let us restate the general theoretical approach for this kind of problem [14]. Consider the following equations for a simple two-dimensional linear system:

$$dx = -\lambda_s x dt \tag{2.10}$$

$$dy = \lambda_u y dt \tag{2.11}$$

where $\lambda_u, \lambda_s > 0$. The trajectories of this system decay in the direction x and grow in the direction y. The passage times depend only on the behavior in a neighborhood of the hyperbolic point $(0, 0)$. Since trajectories leave the box only through the box side $y = \pm\delta$, the x coordinate

does not affect the box passage time. It is necessary to determine the probability function for the time that the orbit stays in the square box with sides of length 2δ. This probability function will depend on the distribution of initial values y_0 with which the trajectory enters the box. The solution for equation 2.11 can be obtained by simple integration:

$$y = y_0 e^{\lambda_u t} \tag{2.12}$$

When $t = 0$, the trajectory enters the box at y_0. Thus the trajectory leaves the box when $|y| = \delta$ at time

$$t = \frac{1}{\lambda_u} \ln |\frac{\delta}{y_0}| \tag{2.13}$$

This formula relates passage time to the initial entry in the box.

It is obvious that the distribution of t is higher for some particular region Δy when $\frac{\Delta t}{\Delta y_0}$ is low. Applying the limit we obtain

$$P(t) = (\lim_{\Delta y \to 0} \frac{\Delta t}{\Delta y_0})^{-1} = |(\frac{dt}{dy_0})^{-1}| = |\frac{dy_0}{dt}| \tag{2.14}$$

Thus,

$$P(t) = |\frac{d(\delta e^{-\lambda_u t})}{dt}| = \lambda_u \delta e^{-\lambda_u t} \tag{2.15}$$

The probability for passage times t has been obtained so far for uniformly distributed y_0. Now 2.15 should be normalized so that

$$\int_0^\infty |P(t)| dt = 1 \tag{2.16}$$

Integrating 2.15 yields:

$$\int_0^\infty |P(t)| dt = \lambda_u \delta \int_0^\infty e^{-\lambda_u t} dt = -\delta e^{-\lambda_u t}|_0^\infty = \delta \tag{2.17}$$

So

$$\int_0^\infty |P(t)| dt = 1 \Longrightarrow \delta = 1 \tag{2.18}$$

Thus, the final expression for the probability is

$$P(t) = \lambda_u e^{-\lambda_u t} \tag{2.19}$$

This result is valid for uniform distribution of y_0. In order to derive a formula for an arbitrary distribution of y_0, the following approach should be taken. Let the variable r represent a physical characteristic of a system. The initial interval is defined in which the values of the variable lie. A transformation of the form

$$s = T(r) \tag{2.20}$$

produces a result value s for every r. It is assumed that the transformation function given in Equation 2.20 satisfies the conditions:

- $T(r)$ is a single–valued and monotonically increasing or decreasing in the interval $0 \leq r \leq \delta$;
- $0 \leq T(r) \leq \delta$ for $0 \leq r \leq \delta$.

The variable r may be viewed as randomly distributed over the interval $[0, \delta]$. If r and s are continuous variables then they may be characterized by their probability density functions $p_r(r)$ and $p_s(s)$ respectively. From elementary probability theory, if p_r and $T(r)$ are known and $T^{-1}(s)$ satisfies the above–mentioned conditions, the probability density function of s is

$$p_s(s) = (p_r(r)\,\frac{dr}{ds}) \tag{2.21}$$

where $r = T^{-1}(s)$. Now it is easy to obtain the probability formula for the situation where y_0 is not distributed uniformity. It may be Gaussian, Poisson or any other kind of distribution. In any case it is possible to use Bayes formula for calculating the final probability

$$P(t) = P(y_0)\,P(t|y_0) \tag{2.22}$$

If a Gaussian distribution is chosen for y_0, then

$$P(t) = \frac{1}{\mu\sigma} e^{-y_0^2+\mu} e^{-\lambda t} \tag{2.23}$$

where σ is the standard deviation and μ is the mean of a random variable.

2.8 Application of Passage Time Theory

The focus of this thesis is to develop empirical techniques for mean passage times distribution in a box around a saddle point. In a real physical system such as a flame it is difficult to analyze spatio–temporal patterns in order to extract the information necessary to apply the same methods that are applied to numerical models. The main task of analyzing intermittency in flame front images can be broken up into several pieces. Borders (or frame numbers) should be established designating where the flame reorganizes itself into another state. Basically it is most important to establish the duration that the system has a stable state. It will be further used in building a histogram of the probabilities for flame stable state durations. This histogram is important because it will show that the flame system is governed by the same laws which were established when analyzing two–dimensional models. Also, the system can be subject to a different kind of noise, white or colored. It is worth mentioning that such experiments were carried out and the system's behavior was observed for changing topological structure, depending on the level of the applied noise. No significant changes were found, which indicates the presence of symmetry in the system. Also the application of noise gives a way to apply Stone-Holmes theory. Note that the noise did not cause any significant change in the unstable eigenvalue. Thus, by applying the **least square method** to the histogram the unstable eigenvalue can be calculated, which allow researchers to predict the system's future behavior.

In this Chapter it is shown how Stone–Holmes theory is applied to a numerical system. In Chapter 3 numerical experiments over Duffing and Ornstein–Uhlebeck models are carried out to show how the passage time of the heteroclinic trajectory in the neighborhood of the saddle point depends on the entry point into a box surrounding the saddle point. It is also shown that Stone–Holmes theory can be effectively applied to models. Chapter 4 contains a description of the main research done in this thesis. It provides algorithms and techniques that allow a researcher to detect heteroclinic connections and build the probability histograms in a real physical dynamic system. In Chapter 5 experiments are described that shown the applicability of

the techniques and algorithms proposed in Chapter 4.

Chapter 3

Analysis of Numerical Experiments

3.1 Overview of Numerical Experiments

For the purpose of this work it is important to find a relatively simple numerical model to test various assumptions, and to see how the model's behavior is influenced by external conditions such as noise. The first objective of the numerical experiment is to show how the passage time of the heteroclinic trajectory in the neighborhood of the saddle point depends on the entry point into a box surrounding the saddle point. The size of the box can be chosen arbitrarily and the only condition is that the trajectory inside the box must be exponential. The second objective of the experiment is to see if Stone-Holmes theory can be applied to the model. To do that a Wiener noise process has to be simulated, and applied to the solution obtained by integrating an Ornstein–Uhlenbeck process equation. The goal of the experiment is to provide sufficient information to obtain an unstable eigenvalue of the system which could then be used to predict the system's behavior. By applying Wiener noise, the deviation of the unstable eigenvalue can be viewed as dependent upon the level of the noise. It is also important to see how noise may affect the histogram of passage times and to compare the results with the theoretical predictions. The passage time definition was given in Chapter 2.

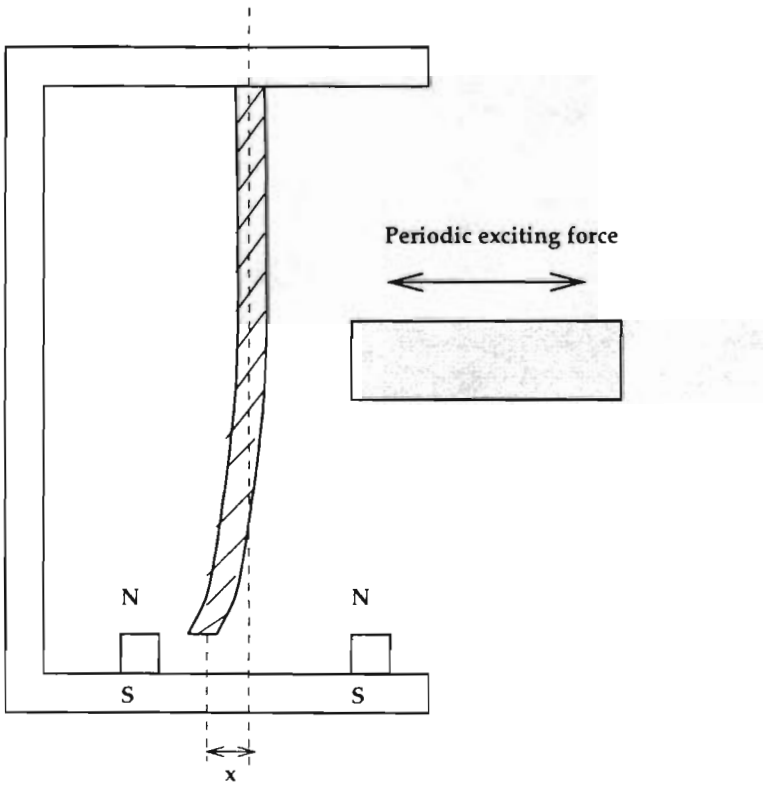

Figure 3.1: Steel beam experiment.

3.2 Description of Duffing Model

Duffing [11] introduced a nonlinear oscillator with a cubic stiffness term to describe the hardening spring effect observed in other mechanical problems [15]. The resulting class of equations are among the most intensively–studied examples in nonlinear physics. Moon [16] considered the Duffing system with external periodic force applied to the system. The experimental setup for this system is shown on Figure 3.1. A slender steel beam is clamped in a rigid framework upon which magnets are installed. Their attractive forces overcome the elastic forces that would otherwise keep the beam straight. Considering the position of the magnets, and consequently the energy well created by the magnetic forces, the beam settles with its tip close to

the energy well. The simplest model equation can be deduced by assuming a potential to be the symmetric one-dimensional field

$$V(x) = \frac{x^4}{4} - \frac{x^2}{2} \tag{3.1}$$

As shown in Figure 3.1, x represents the coordinate of the tip displacement from the zero position which is the normal position of the tip in the stable situation. Generally, x is a characteristic measure of the position of the control element (beam). The force acting on the beam is governed by the gradient of V. Thus, by applying Newton's second law it is possible to write down the mathematical description of the model

$$\ddot{x} = -grad\ V. \tag{3.2}$$

Substituting V from Equation 3.1 into Equation 3.2 results in the equation

$$\ddot{x} - x + x^3 = 0. \tag{3.3}$$

The next step is to consider dissipation due to different mechanical factors like friction or viscous damping from the surrounding air. Let's say that the dissipation is proportional to the linear velocity. A measure of relative strength of the magnetic and electrical forces can also be added. Then Equation 3.3 can be written as

$$\ddot{x} + \delta\dot{x} - \beta x^2\dot{x} - x + x^3 = 0 \tag{3.4}$$

where δ is the coefficient of viscous damping and β is the amplitude of the strength of magnetic forces. Equation 3.4 can be rewritten as a system of first order ordinary differential equations (ODEs).

$$\begin{cases} dx = & ydt \\ dy = & (x - x^3 - \delta y + \beta x^2 y)dt \end{cases} \tag{3.5}$$

The behavior of the system can be represented by a phase portrait plotting x versus y using time as a parameter. The phase portraits for Equation 3.5 are well–known [7] and shown in Figure 3.2 and Figure 3.3. With $\delta = 0$ the phase portrait has only one fixed point. If $\delta > 0$, then there

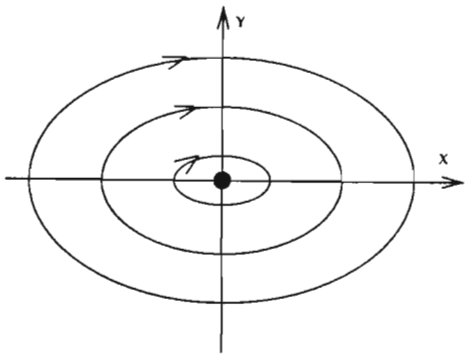

Figure 3.2: Phase space of Duffing ODE with $\delta = 0$.

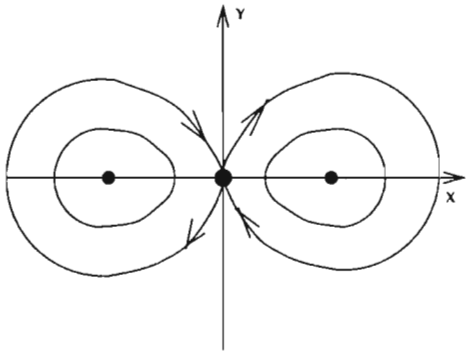

Figure 3.3: Phase space of Duffing ODE with $\delta > 0$.

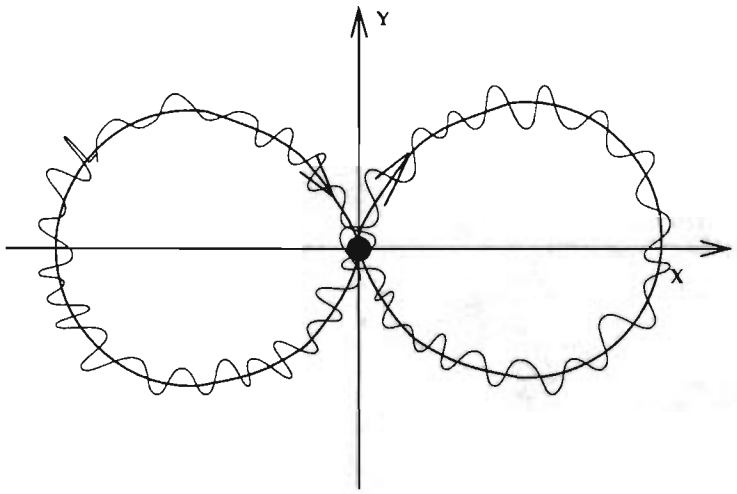

Figure 3.4: Homoclinic trajectory of Duffing ODE with Wiener noise.

are three fixed points as shown in Figure 3.3. The fixed point $(0, 0)$ becomes hyperbolic, so the mean passage time can be calculated by constructing a box around the point and measuring time during which the trajectory stays in the box.

When Wiener noise is applied to Equation 3.5. The system becomes

$$\begin{cases} dx = ydt + \varepsilon dW_x \ , \\ dy = (x - x^3 - \delta y + \beta x^2 y)dt + \varepsilon dW_y. \end{cases} \tag{3.6}$$

where W_x and W_y are the constituents of Wiener process applied to the equation with deterministic parts and $\varepsilon \in [0, \infty)$ is its amplitude. The Wiener process has a zero mean and expectation [1]

$$\frac{1}{2}\langle (dW_x \pm dW_y)^2 \rangle \ = \ \frac{1}{2}\langle (dW_x^2 + dW_y^2) \rangle \ = \ dt \tag{3.7}$$

A qualitative description of the homoclinic trajectory with the applied Wiener noise is shown Figure 3.4. The resulting trajectory, depicted by a thin line, stays near the homoclinic orbit of the noise free system, which is shown by the dark figure eight in Figure 3.4. If we construct an imaginary window having coordinates $(-a, b), (-a, -b), (a, -b), (a, b)$ as shown in Figure 3.5,

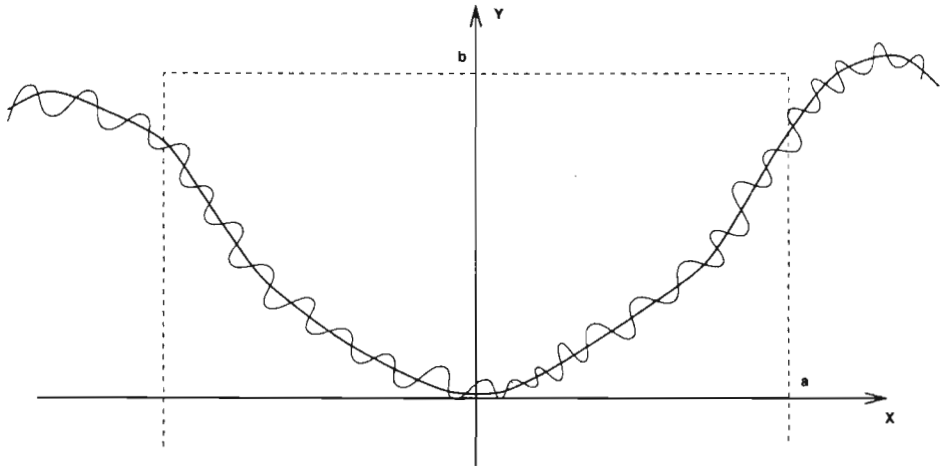

Figure 3.5: Schematic of the homoclinic orbit with applied Wiener noise.

then the trajectory can be seen as a graphical representation of the Ornstein - Uhlenbeck process.

From this point a Stone–Holmes theory is to be applied to the Ornstein–Uhlenbeck process. Noise should be applied to all of the trajectory points that are calculated.

3.3 Computational Algorithm for the Duffing ODE

Three problems are investigated during the numerical experiment. First, it has to be determined whether the distribution of trajectory passage times for different distributions of initial conditions on the vertical boundary of an imaginary box is exponential. An analytical expression for the distribution can be found by using a least–square fitting algorithm.

Second, the numerical integration of Duffing equation with applied noise has to be carried out. The algorithm may be outlined by mentioning that the equation has to be numerically integrated and the integration results at each step are corrected by a value produced by a noise generator. A distribution of passage times is accumulated and parameters for Stone-Holmes formula are calculated.

Third, the second experiment is repeated with different amplitudes of Wiener noise

level. The effect of varying the noise amplitude is analyzed. It may lead to a conclusion on how noise affects the final distribution of passage times, and consequently determine whether or not it is valid to use the unstable eigenvalue as a major descriptive parameter when the noise in the physical system is not controlled.

3.3.1 Main Program Cycle

A software package was developed in X–windows to investigate these issues. The main diagram of the program's flow is shown in Figure 3.6. In order to make the accurate analysis required by this experiment, a huge array of data must be accumulated. The main supplier of the data is the ODE Solution Engine. It takes initial values for an ODE being solved and uses a numerical algorithm to produce the solution for the next time step. The output data are the model time t_i, output value x_i and its derivation $\dot{x}_i$ where i is an integration step number. Since an Ornstein–Uhlenbeck process is simulated, Wiener noise must be applied at every step of the solution. As seen on the diagram there is a separate Wiener Noise Engine that is running in parallel to the ODE Solution Engine. At every step noise correction values are multiplied by an amplitude in the Wiener Noise Amplitude Multiplier and added to the output data x and $\dot{x}$ of the ODE Solution Engine. The corrected data is passed to the Passage Time Generator which generates a passage time value for the current loop of the trajectory through the boundary box. These passage time values are stored in a Data Accumulator that is a binary file. The program returns to the Engine and runs in the loop unless an Initiator, which is a user interface button, sends a signal to analyze the accumulated data. The data are supplied to the Histogram Builder that sorts the data and calculates the probabilities. The histogram probability data is supplied to the Adaptive Least–Square Fitting Procedure that calculates an unstable eigenvalue for the current data set.

is bigger than the step of integration (and it normally is), then it should be normalized. Another random number is generated, and the divisor = max(random_number1, random_number2) is found. The smaller number is divided by the divisor and result is smaller then 1. The resulting value is divided by 2 until it becomes less then the step of integration. This normalization generates Wiener noise components that satisfy equation 3.7 defined in [1].

In order to test the validity of this algorithm for generating Wiener noise a test program was created. An essential part of this program is to simulate Brownian motion by using the Wiener noise algorithm. The standard algorithm for producing Wiener noise is modified in this thesis. Every time a new Wiener noise number is calculated, it is multiplied by -1 making it change its sign at every step. The reason for this is to control the stability of the heteroclinic orbit by forcing the orbit with the noise to fluctuate around the stable trajectory. The starting point for the noise process is depicted in the center of a circle of radius 1 as shown in Figure 3.7. Generally speaking, the radius is equivalent to the step of integration. The result is shown this way to see how much and how often the algorithm produces collinear motion vectors that contribute mainly to one direction. The idea is to check whether the algorithm produces random zero mean motion vectors. The expected resulting motion in one case is schematically shown in Figure 3.7 with connected segments of lines. In the box experiment, both components of the Wiener process are multiplied by an amplitude value. The amplitude allows the analyst to manipulate the noise level, an essential part of this experiment. It should be noted that at every new loop of the calculation, the W components should change sign so that the definition of zero mean of Wiener process will hold in this experiment.

In order to test the proposed algorithm for a Wiener noise generator, a test program has

3.3.2 Wiener Noise Generator

As mentioned in Section 2.4, Wiener process is another name for Brownian motion. The Wiener Noise Generator is a very simple program. Its flow chart is shown in Figure 3.8. A standard `rand()` function from the **C** library is used to generate a random number. If this number is bigger than the step of integration (and it normally is), then it should be normalized. Another random number is generated, and the divisor = max(random_number1, random_number2) is found. The smaller number is divided by the divisor and result is smaller then 1. The resulting value is divided by 2 until it becomes less then the step of integration. This normalization generates Wiener noise components that satisfy equation 3.7 defined in [1].

In order to test the validity of this algorithm for generating Wiener noise a test program was created. An essential part of this program is to simulate Brownian motion by using the Wiener noise algorithm. The standard algorithm for producing Wiener noise is modified in this thesis. Every time a new Wiener noise number is calculated, it is multiplied by -1 making it change its sign at every step. The reason for this is to control the stability of the heteroclinic orbit by forcing the orbit with the noise to fluctuate around the stable trajectory. The starting point for the noise process is depicted in the center of a circle of radius 1 as shown in Figure 3.7. Generally speaking, the radius is equivalent to the step of integration. The result is shown this way to see how much and how often the algorithm produces collinear motion vectors that contribute mainly to one direction. The idea is to check whether the algorithm produces random zero mean motion vectors. The expected resulting motion in one case is schematically shown in Figure 3.7 with connected segments of lines. In the box experiment, both components of the Wiener process are multiplied by an amplitude value. The amplitude allows the analyst to manipulate the noise level, an essential part of this experiment. It should be noted that at every new loop of the calculation, the W components should change sign so that the definition of zero mean of Wiener process will hold in this experiment.

In order to test the proposed algorithm for a Wiener noise generator, a test program has

Figure 3.7: Schematic doodle of Brownian motion.

been written. The main idea of this program is to illustrate the concept of Brownian motion as it might happen in real life. Each output of the Wiener noise generator is simulated with a virtual particle. A line connects a previous position of the particle with a new position. The scales for the output screen are chosen to create a visual border for the virtual particle. The program has been written in **C++** and run under **Windows NT**. The program used a timer that called the Wiener noise generator after the specified period of time. The resulting motion for one of the experiments is satisfactory for the definition of Brownian motion.

3.3.3 Passage Time Calculation

The algorithm for a passage time calculation can be decomposed into two main features: obtaining precise entry time when a heteroclinic trajectory enters the box and fixing the exit time when it leaves the box.

An important question is what size should the box be chosen to obtain correct results for the Duffing equation. The size of the box should be relatively small, let's say $\frac{|Box_x|}{\max(|T_x^+|)} \ll 1$

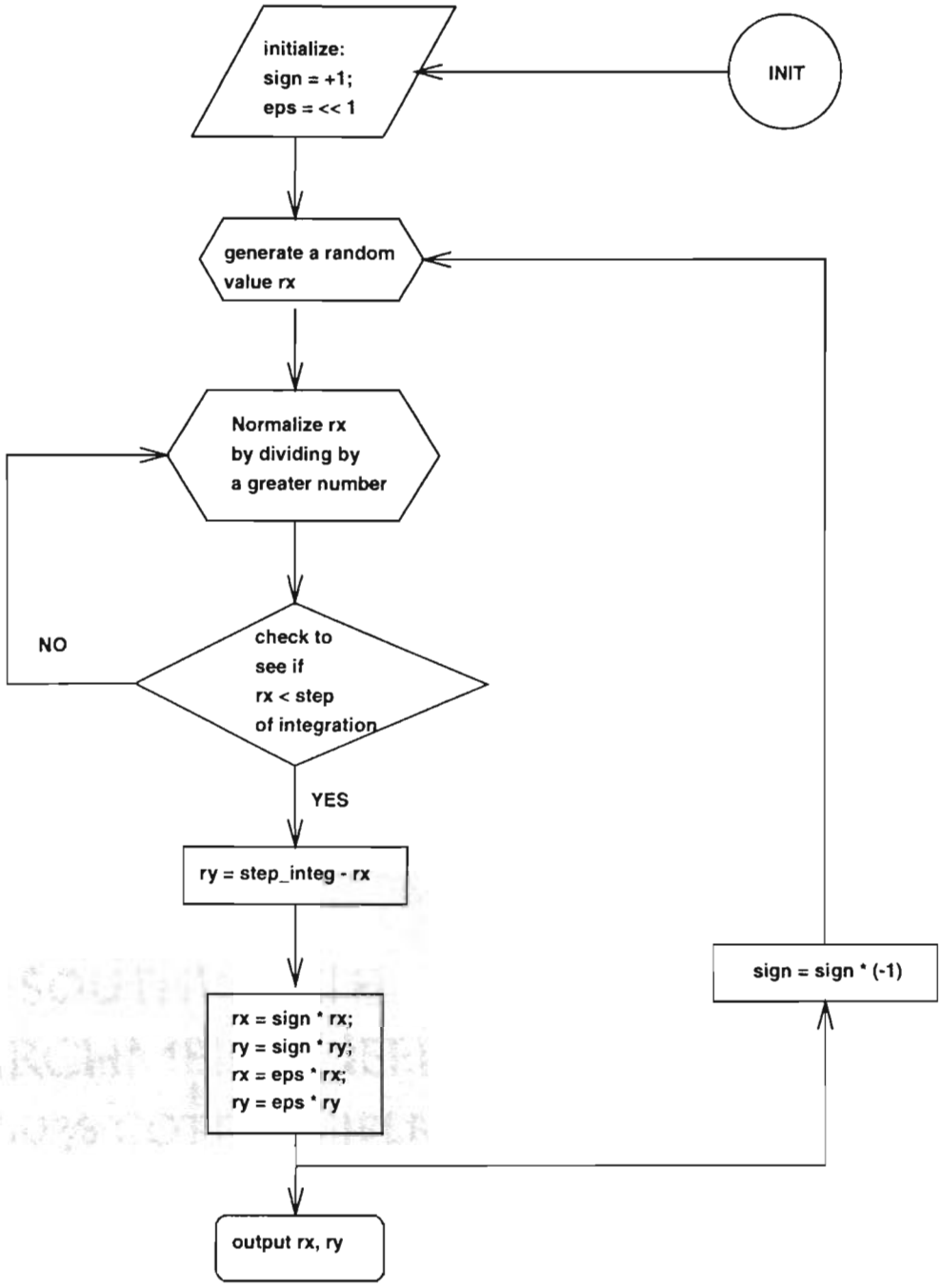

Figure 3.8: Wiener noise generator.

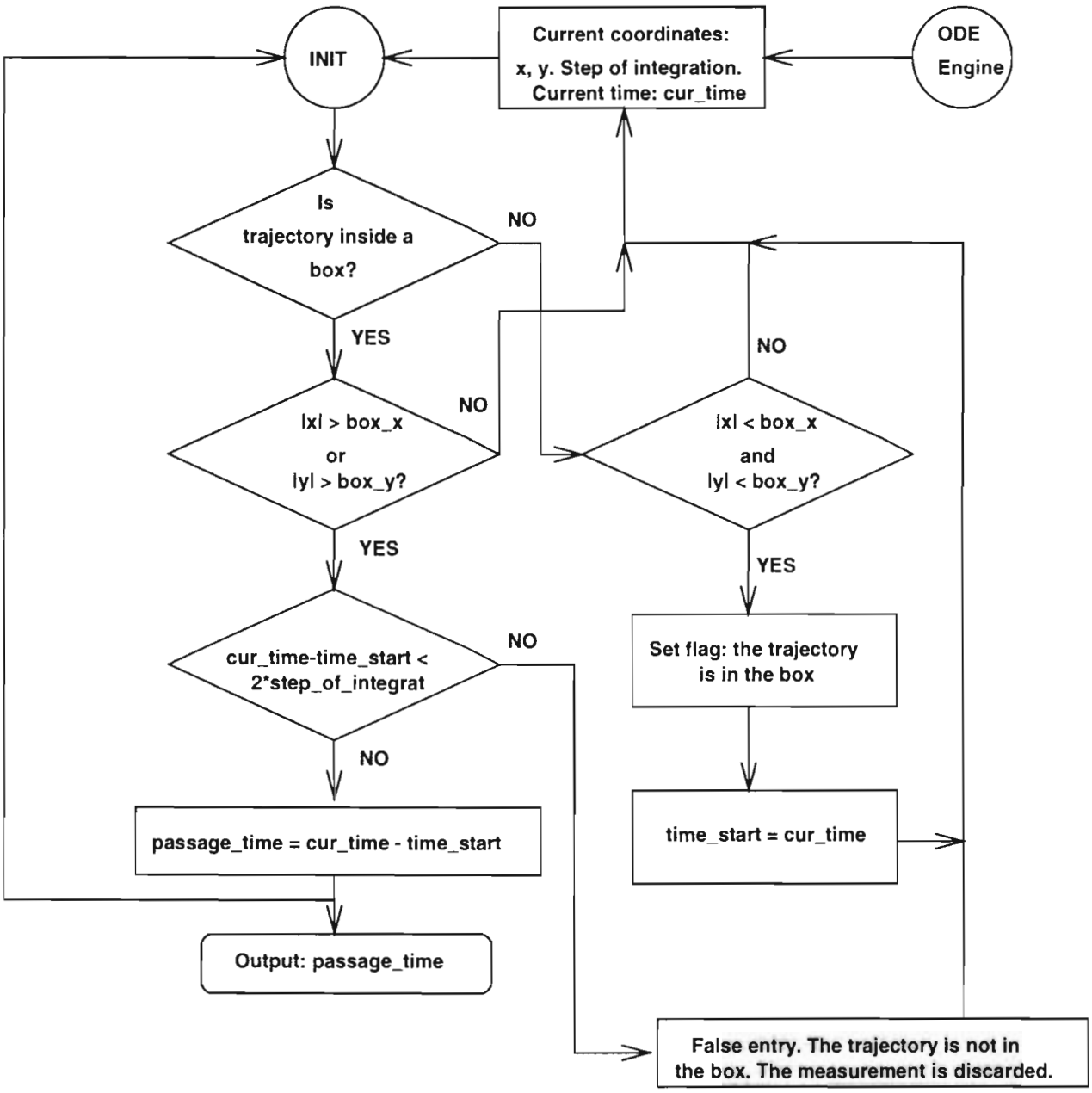

Figure 3.9: Box entry and exit points calculation algorithm.

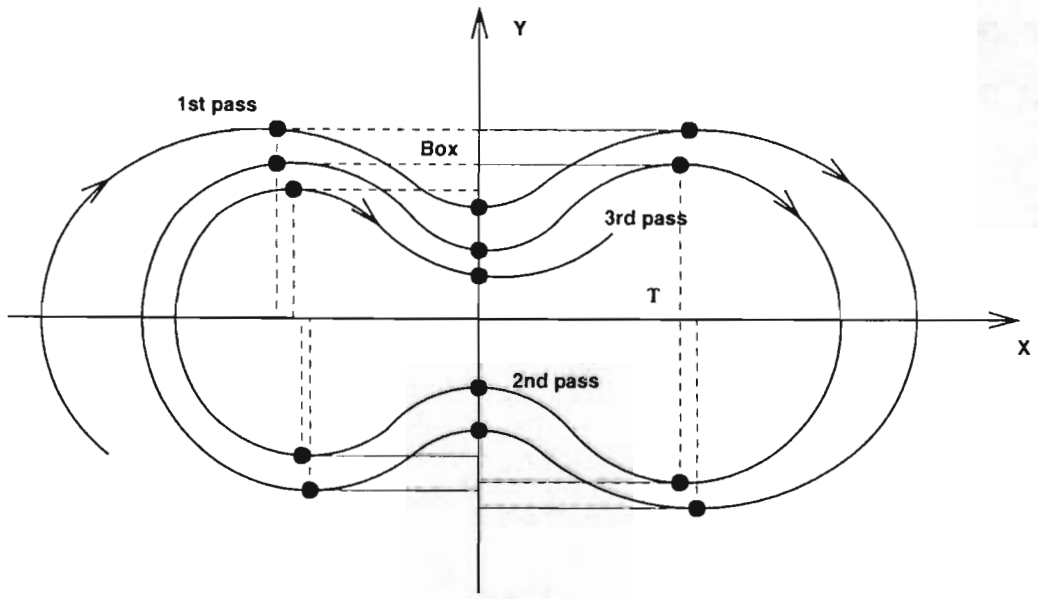

Figure 3.10: Construction of a box around a hyperbolic point.

and $\frac{|Box_y|}{\max(|T_y^i|)} \ll 1$ where $|T_x^i|$ is the modulus of the x coordinate at which the first derivative of x at pass i is equal to zero, and Box_x, Box_y are the lengths of the sides of the box along x and y axes respectively. It should be pointed out that these inequalities are obtained experimentally and represent the qualitative estimate that a researcher should use when he picks up the size of the box. For the further use let's introduce the definition of *climax*. *Climax* is the modulus of the x or y coordinate of the point at which the first derivative of x at a pass is equal to zero. By applying the max operation the maximum value of x can be found in the vector of all values T_x^i. A similar definition holds for the y coordinate. This concept is shown in Figure 3.10.

The algorithm introduces a flag variable checking to see whether a trajectory is inside or outside the box. If the trajectory is outside the box, then at every calculation step, values x_i and y_i are checked to see whether the trajectory has entered the box. There is a caveat: since Wiener noise is applied, a false entry may be present. It means that the trajectory may enter the box at one step of integration and leave it at the next step. A simple criteria is used in the

algorithm shown Figure 3.9. If a trajectory stays for two steps of integration after it entered the box then the entry is assumed to be true. At the moment of entry, the start time t_i is saved. When a trajectory leaves the box, the end time t_j is obtained. The passage time is calculated as passage_time $= t_j - t_i$.

3.3.4 ODE Numerical Solution Algorithm

A standard Runge–Kutta algorithm with adaptive stepsize control [17] is used to solve the ordinary differential equations. The purpose of adaptive stepsize control is to achieve some predetermined accuracy in the solution with minimum computational effort. It is possible to make an analogy with land topology which has meadows and smooth countryside along with treacherous terrain. One can assume that the treacherous terrain requires smaller steps than traversal of meadows. In the context of ODEs the treacherous terrain occurs at times during which the solution is varying rapidly. The algorithm is illustrated in Figure 3.11. The algorithm checks calculated error values and compares them with minimum and maximum value boundaries. If the error exceeds the maximum boundary then the stepsize of integration is halved and integration is repeated. If the error becomes smaller the minimum, the stepsize is multiplied by two.

3.3.5 Passage Time Histogram Building Algorithm

The purpose of the histogram building algorithm is to produce a probability density for the passage times. The algorithm is shown in Figure 3.12. The array of passage times is scanned first to find maximum and minimum passage time values. The histogram range is the difference between the maximum and minimum values. The histogram range is divided into intervals. There is no theory explaining how to discretize the histogram range. Picking very small or very big intervals may affect the final result. In an experimental data set there is a finite number of points. If a number of bins in the histogram is comparable to the number of points, then the

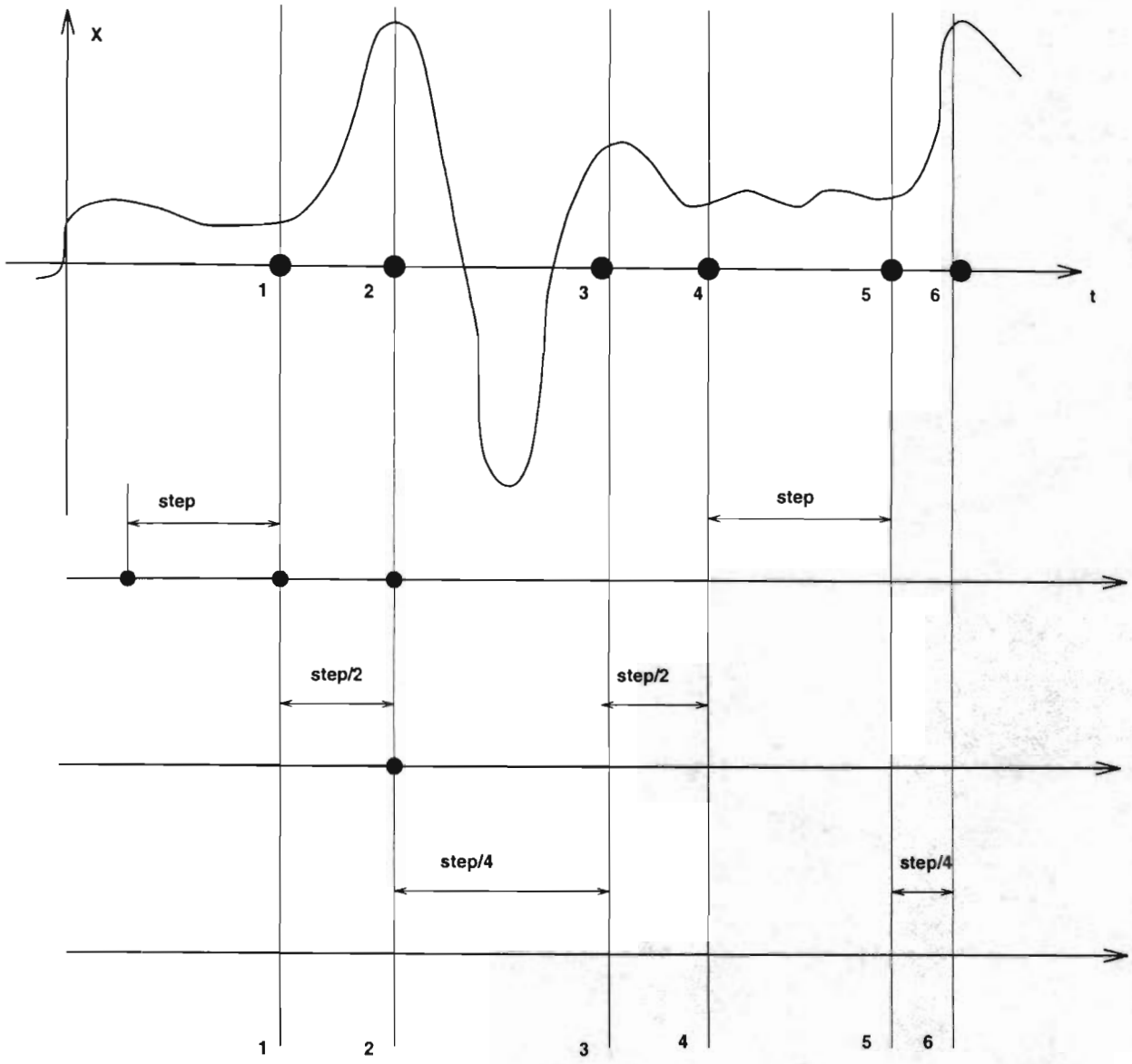

Figure 3.11: Adaptive stepsize control in Runge-Kutta algorithm.

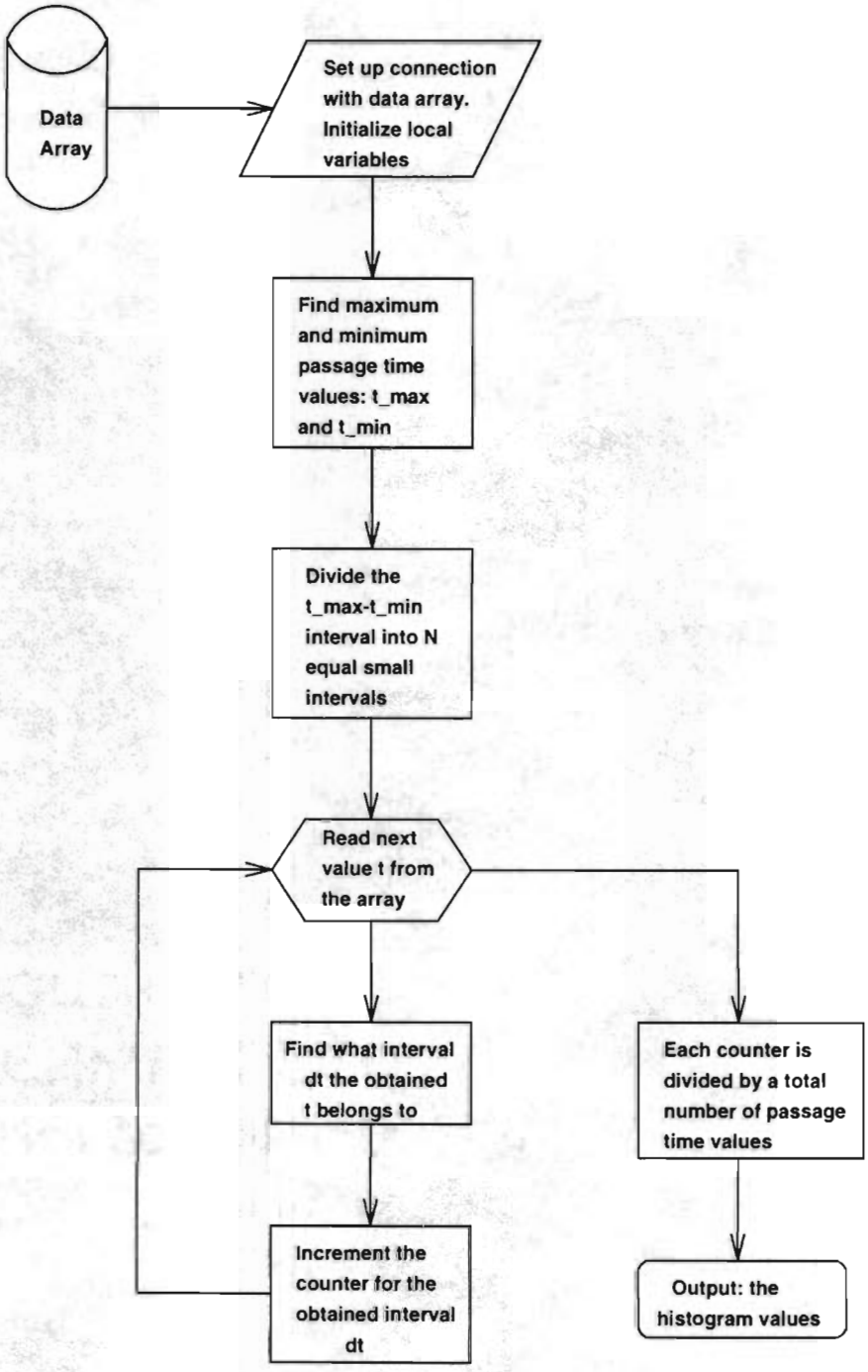

Figure 3.12: Passage Time Histogram Building Algorithm.

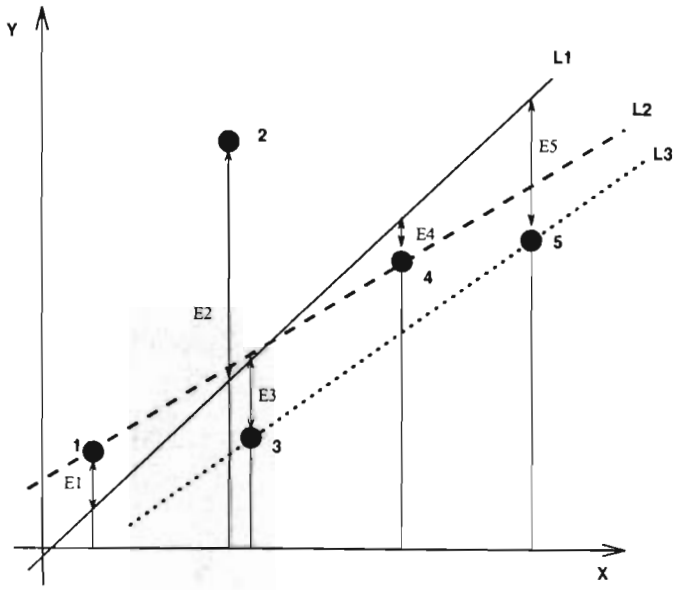

Figure 3.13: Least-Square Fitting with Adaptive Adjustment.

histogram will look unnaturally skewed. The optimal solution is found by trying different values and evaluating the resulting histogram to find out whether it has exponential fall–off. Thus, after dividing the histogram range into intervals, each interval is assigned a counter. The counter is incremented when a passage time value falls into the corresponding interval. After all passage times have been read, each counter is divided by the total number of passage time values and the probability for the passage time to get into each interval is found.

3.3.6 Adaptive Least–Square Fitting

A least–square fitting algorithm [17] is modified to analyze the behavior of the histogram. The general idea of the adaptive least–square fitting algorithm is illustrated by Figure 3.13. If the points are grouped in clusters according to their spatial location, then the majority of points P_i may lie around some line. Since the final purpose of the experiment is to obtain

an unstable eigenvalue it is reasonable to neglect outliers because of the presence of Wiener noise.

There are five points depicted in Figure 3.13 that are designated $\{1, 2, 3. 4. 5\}$. If a line L1 is drawn, then errors $\{E_1, E_2, E_3, E_4, E_5\}$ for each point can be obtained as an indication of the accuracy of this line fitting. It is known that the result of the experiment can be influenced by noise. Initially, using this point of view, an equal probability can be assigned that each point has the biggest error. Another assumption can be made about the maximum number of points that can be largely influenced by noise. The assumption says that the number of points that are taken as input for the least–square fitting algorithm must be bigger than the number of points discarded as having a big noise error. In other words we say that more than half of all the points should remain after discarding some points that are determined to have the biggest deviation from the best line fit. Thus, the main idea of adaptive least–square fitting consists of determining the points that contribute the largest calculation error and discarding these points. A standard least–square algorithm is applied to fit an equation of the form $y_i = ax_i + b$ to the data. So after obtaining coefficients a and b for the data array containing N points, for each point x_i new value $y_i = ax_i + b$ is calculated. The error between the calculated value and empirically obtained value is $E = |y_{e_i} - y_i|$. The empirical point having the largest error is discarded and new coefficients are calculated. The algorithm stops when the new coefficients differ from the recent old ones by no more then ten percent. Let's take a look again at Figure 3.13. In the beginning the standard least–square fitting is applied to the full set of points. Line L1 is the result.

Notice that $E_2 > E_i$ where $i = 1, 3, 4, 5$. One could say that if this point was discarded in the very beginning then there would not be any necessity to spend time on this calculation. Now imagine that there was never point E_2 present in the data set. Still it is quite unclear by looking at lines L2 and L3 whether point E_5 or point E_3 must be discarded.

As a check, the optimal fit of 4 of 6 data points applies standard least–square fitting algorithm in $M = \frac{N}{2} - 1$ steps where N is the number of points in the data set. At every step the point having maximum error should be discarded.

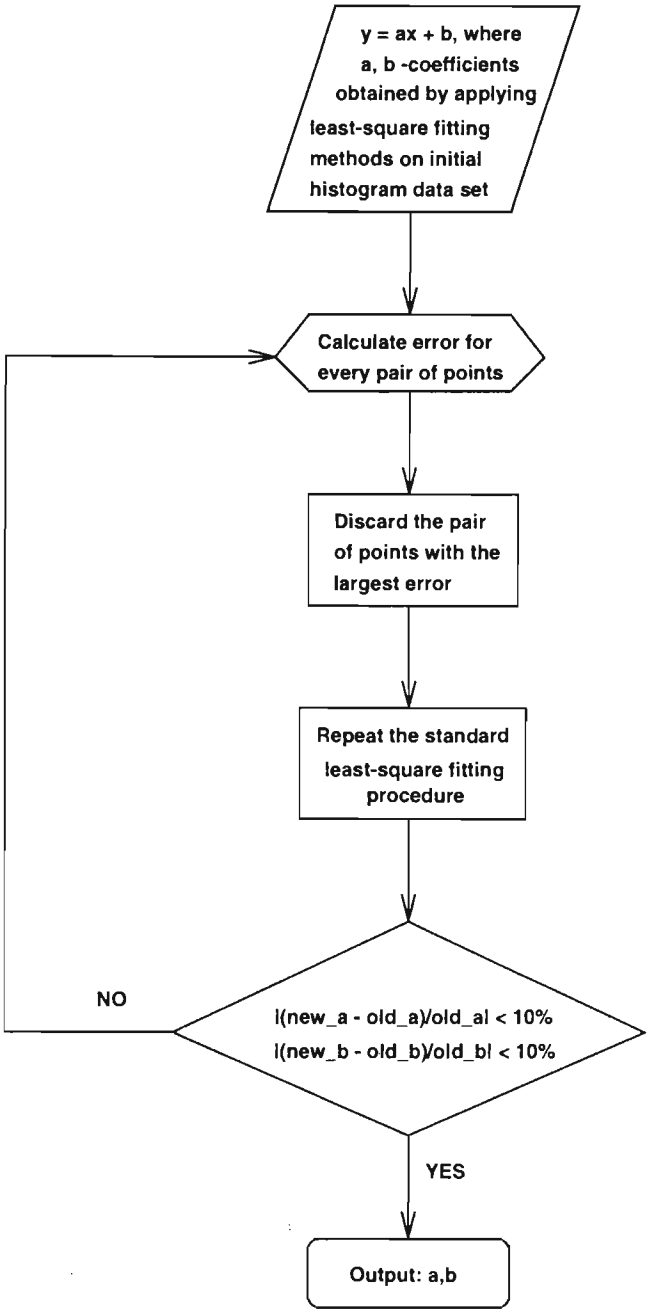

Figure 3.14: Adaptive least–square fitting algorithm.

Another verification is that the coefficients are obtained by taking all possible pairs (x_i, y_i) and (x_j, y_j) where $i \neq j$. For six points there will be $C_2^6 = \frac{6!}{2!(6-2)!} = 15$ combinations of the coefficients. Comparing one pair to another the optimal solution was found and agreed with the value obtained by adaptive fitting in the cases tested.

3.4 Description of Software

In order to facilitate the exploration of dynamic systems with noise, the software programs called **LinOde** and **HeteroCon** were created.

HeteroCon's architecture contains computational simulation module shown in Figure 3.6. The Graphic User Interface (GUI) contains two submodules. The first submodule is responsible for displaying various graphs describing the system's behaviour. It accepts mouse input and allows the analyst to select a part of a graph to zoom in or out. It allows the analyst to enter coordinates of a box in which the solution is presumed to stay linear. The second submodule is responsible for providing the textual input and output of control parameters. It also allows the analyst to launch a computational module and to stop its execution.

In Figure 3.15 and Figure 3.16 a picture of the GUI can be seen, showing the graphs of the phase plane and time series of the Duffing ODE respectively. A reseacher can interact with the daemonized process by using six control buttons on the front panel. They are: **Phase**, **Time**, **MPT**, **Prob**, **Setting**, and **Quit**. The **Phase** and **Time** buttons serve to show the graphs of the phase plane and time series at the run time of the daemonized program or statically. The **MPT** button is used to show the graph of passage times. A researcher may need it to see how the passage time changes when calculating the next output values, and it may be a good debugging tool for detecting possible false box entries. The **Prob** button is used to show the histogram of the distribution of the probabilities of a passage time value to be in a defined interval. The **Setting** button brings up a window with fields enabling editing ofthe following parameters: *initial time* t_0, *initial coordinate* x_0, *initial derivative* $\dot{x}_0$, *stepsize of integration*, *Duffing ODE parameters*

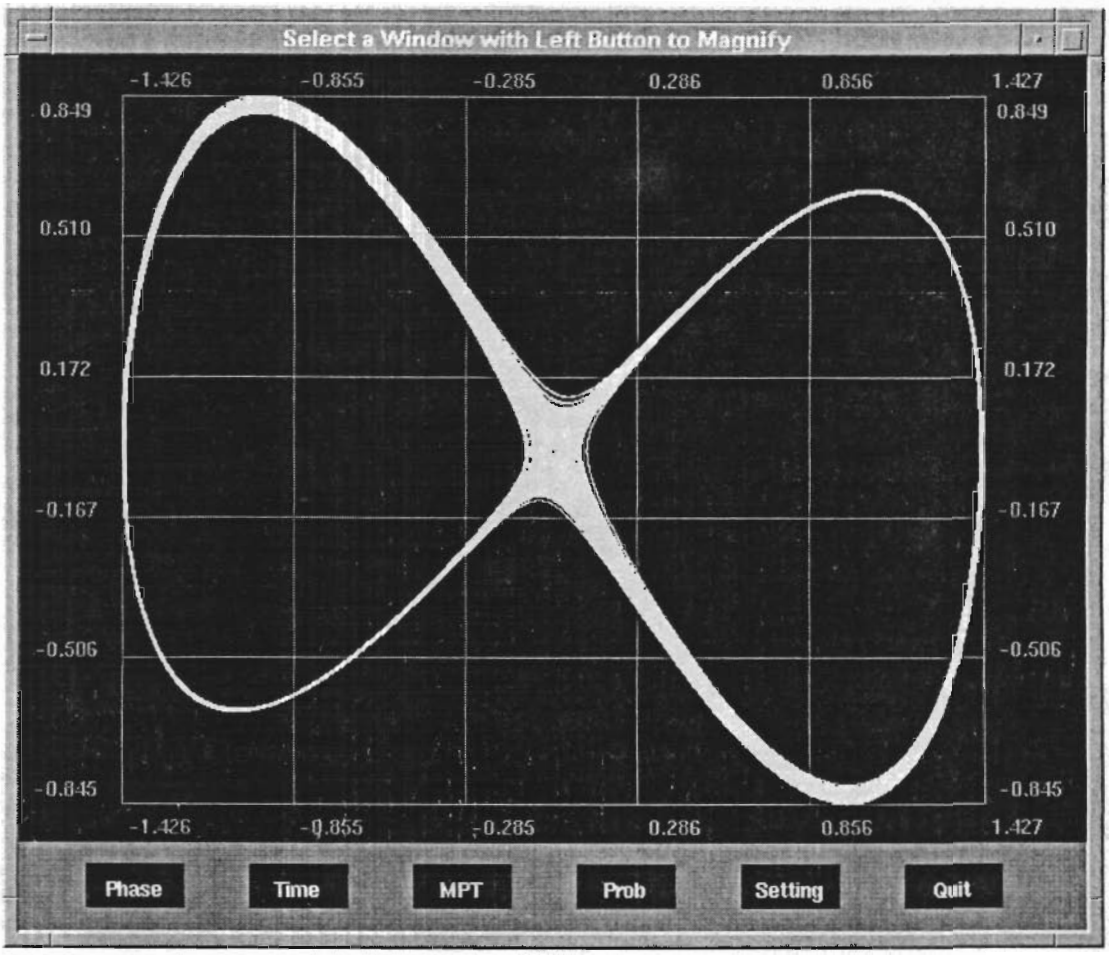

Figure 3.15: Phase space of Duffing ODE.

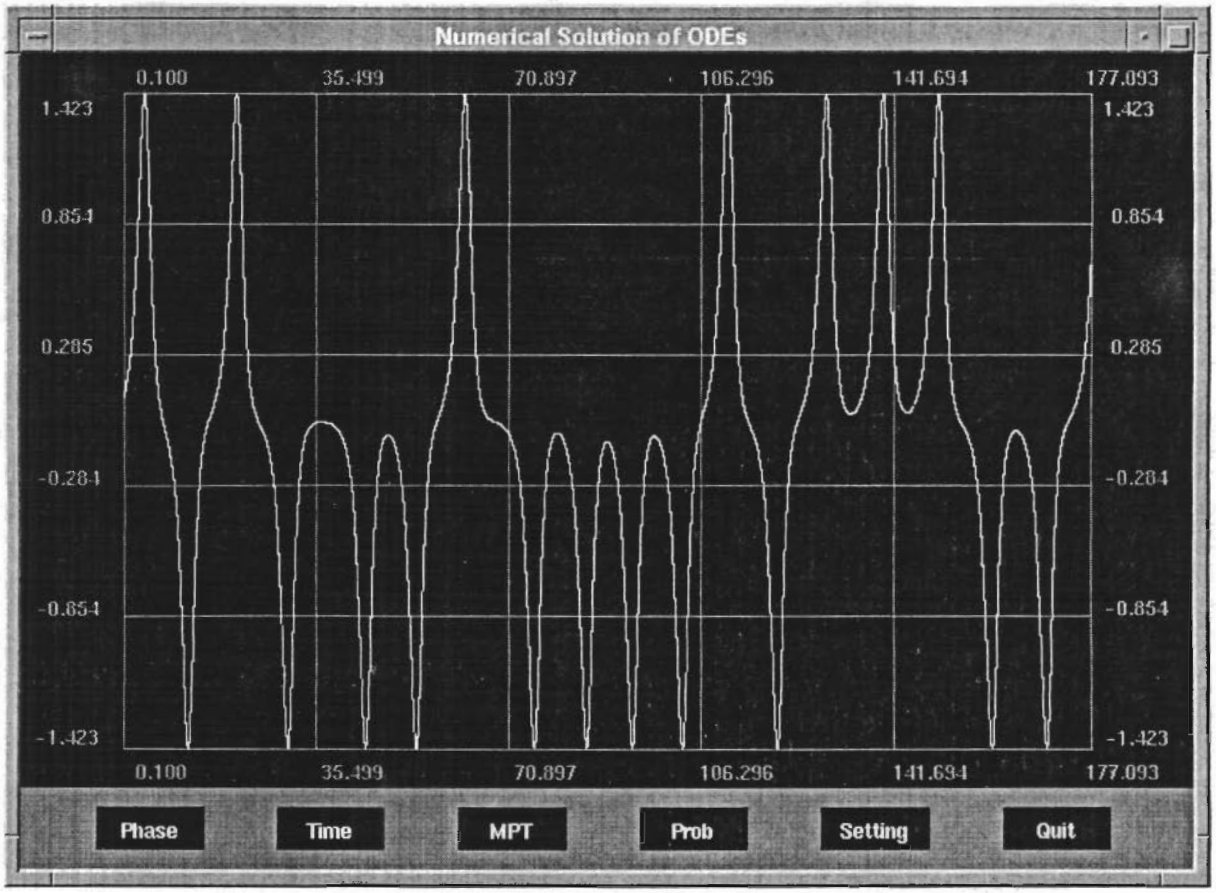

Figure 3.16: Time series of Duffing ODE.

β and δ, *amplitude of the Wiener noise process* ε, and the *file name* for saving data. The window also provides the information fields *number of steps* giving the number of steps executed since the program started, and *current time* showing the current time in the equation relative units. The last button is **Quit**. By pressing it a researcher is able to shut down the main program and all daemonized processes. The setup window has four buttons: **Go**, **Read**, **Stop**, and **Cancel**. The **Go** button starts the daemonized ODE Solution Engine program. If the initial data must be read from the initial data file, and not entered manually, then the **Read** button must be pressed and the initial data will be read from the file if it exists. The **Stop** button allows a researcher to stop the daemonized process without quitting the main program. And finally, the **Cancel** button returns a researcher to the main window.

The Computational module, which runs in daemonized mode, performs the numerical simulation of the model system and analyzes the results. It is running as a separate process and communicates with the GUI by using binary files and shared memory. The central module is the program for solving the ordinary differential equation, using a fourth order **Runge-Kutta** algorithm with adaptive stepsize control. This program checks to see if the solution entered the predefined box and left it. All the data necessary for the following computation are saved in a binary file. When a user sends a message from the GUI that the analysis is over, the accumulated data is analyzed by invoking several programs. The first program builds a histogram from the accumulated data. The histogram shows the dependance of the natural logarithm of the probability of ranges of passage times from the passage times ranges. Then an adaptive least square fitting module is applied to find the coefficients of the best line fitting the set of histogram coordinates. Finally the unstable eigenvalue describing the system's behavior is obtained by exponentiating the line coefficients. A researcher can see any part of a graph zoomed up by clicking the left button of the mouse on the first corner of the window to be zoomed, dragging the mouse with the left button pressed, and then releasing it over the opposite corner of the window. The part of the graph shown in this window will be zoomed up to the extent of the current screen.

3.5 The Box Experiment With the Ornstein–Uhlenbeck System

As explained in the previous chapter, certain characteristics of a complex heteroclinic orbit's behavior may be reduced to an investigation of how the orbit behaves in the neighborhood of the hyperbolic fixed points. Some insight into these issues can be obtained by the study of Ornstein–Uhlenbeck process:

$$\begin{cases} dx = & -\lambda_s x dt + \varepsilon dW_x \\ dy = & \lambda_u y dt + \varepsilon dW_y \end{cases} \tag{3.8}$$

These equations describe a trajectory that decays in x and grows in y. The passage times depend only on behavior in the neighborhood of the hyperbolic point. Since the trajectory leaves the box only through the box side $y = \delta$, the x component does not affect the box passage time. It is necessary to determine the probability distribution for the time the orbit stays in the square box with side equal to δ. As it is already seen, the probability distribution will depend on the distribution of initial values y_0 with which it enters the box. The strict analytical formulas for the system without Wiener noise were obtained in the previous chapters. It is important to verify that resulting probability has asymptotic behavior $P \sim e^{-\lambda_u t}$.

The first experiment is carried out with the Ornstein–Uhlenbeck system. In the straightforward implementation of this experiment the range of the initial values must be chosen as $y_0 = (0, \delta]$. At each step the initial value is incremented by some infinitesimal positive decimal number. The passage time is calculated for each y_0 and used to build a histogram. Trajectories that start with $y_0 = 0$ do not leave the box, since $(\delta. 0)$ lies on the axis x which is a separatrix for the Ornstein–Uhlenbeck system. After a few steps the computation will converge to fixed point $(0, 0)$ that is a stable attractor, so this startup does not have any practical value for the experiment. The starting value y_0 should be chosen small enough so that the resulting data is incomplete, but not so small that it would take large amount of time to pinch out of

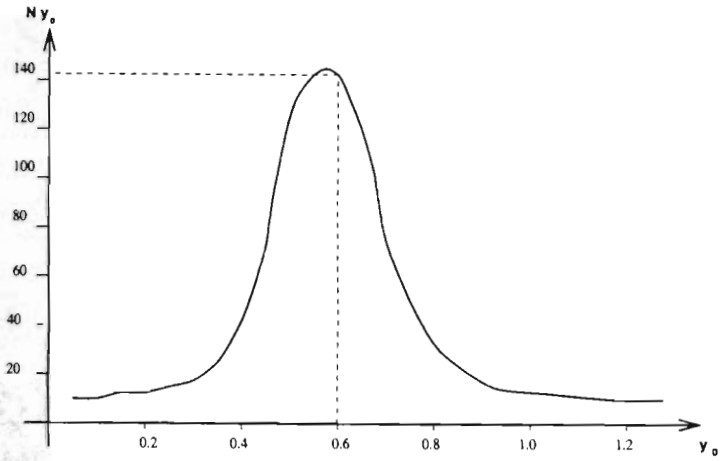

Figure 3.17: Gaussian distribution modelling graph.

the neighborhood of the fixed point. For each experiment, the initial value y_0 was chosen at 10^{-7} and was incremented by 0.005. The resulting histogram was analyzed and the unstable eigenvalue was estimated using the adaptive least–square fitting algorithm. This part of the experiment was done for the uniform distribution of the initial value.

For Gaussian or Poisson distribution of the initial values a simple algorithm was used. For each box size the initial values were chosen using the same rule described for uniform distributions. The passage time calculation for each initial value y_0 was repeated N_{y_0} times. N_{y_0} corresponds to the relative frequency y_0 appears in the chosen distribution as shown in Figure 3.17. It means that calculation of passage times for value y_0 is repeated N times. As it is seen in the Figure 3.17, if $y_0 = 0.6$ then the passage time for this entry in the case of a Gaussian distribution should be put into the resulting array 144 times. If zero Wiener noise amplitude is used then the numerical integration leads to the probability histogram that will subsequently lead to the calculation of an unstable eigenvalue almost equal to the unstable eigenvalue in the Equation 3.8. "Almost" means that there is a small error accumulated while applying numerical methods that give some preset accuracy. After the unstable eigenvalue is obtained for zero Wiener noise level, the noise amplitude is increased and a new unstable

eigenvalue is calculated. The experiment was repeated both for the uniform distribution of the initial value and for the Gaussian distribution. The purpose of this experiment was to evaluate how much the unstable eigenvalue is influenced by Wiener noise.

3.6 The Box Experiment With the Duffing System

The second experiment involves applying Wiener noise to the solution of Duffing's ODE. The control parameters are picked in such a way that the solution is attracted to a stable heteroclinic trajectory within two or three passes. Wiener noise is characterized by its amplitude. By varying the amplitude it is possible to see how noise affects the estimate of the unstable eigenvalue obtained from the passage time calculation. The amplitude in the experiment was changed from 0 to $1.2y_{max}$, where y_{max} is the maximum y coordinate that a trajectory reaches when solving Duffing equation without applying Wiener noise. This maximum amplitude was chosen to see the error in the unstable eigenvalue estimate when such high noise distorts the skeleton shape of the trajectory. Several thing are important to see as results of the experiment. First of all, one should make sure that the resulting histogram has an exponential fall–off. If so, then an unstable eigenvalue can be obtained by applying an adaptive least–square fitting algorithm to the histogram's tail. Also, it may be interesting to see whether the dependance of mean passage time on the amplitude of Wiener noise is an exponential one, as Stone–Holmes theory predicts [1]. The main difference between the Duffing's equation heteroclinic trajectory and the Ornstein–Uhlenbeck orbit is the nonlinearity of the Duffing orbit when moving out of the neighborhood of the hyperbolic point. This nonlinearity causes the Duffing's equation results to depend on the box size, the other interesting aspect of the problem. The adaptive least–square fitting algorithm is used in the experiment. Since the algorithm was never used in such a context, it may be useful to investigate the properties of the algorithm working on real experimental data. It is particularly important to see whether it tries to get the best of the input data set and does not discard data that may be the essential constitutient of the unstable eigenvalue with a high

degree of accuracy. So the output of the experiment should be a graph comparing the set of unstable eigenvalues obtained by using the full set of experimental probability data versus a data set selected automatically by the adaptive least–square fitting algorithm.

3.7 Results

The first experiment with the Duffing model with Wiener noise applied was carried out with $\beta = 0.498, \delta = 0.4, x_0 = 0.2, \dot{x}_0 = 0.3, \varepsilon = 0.25$. These values were chosen based on calculations made by Stone [1] using a Melnikov perturbation method to guarantee that the system has a heteroclinic trajectory. Table 3.1 shows the resulting probabilities for two experiments using different seeds for the randomly generated data. The first column of Table 3.1, "Passage Time Interval", shows the values of the right border of each histogram's subinterval. The first interval starts with zero and goes to 0.0466 time units. The second column, "Probability", is divided into two subcolumns. The first subcolumn displays the calculated probability for the first experiment. The second subcolumn displays the calculated probability for the second experiment. And finally, the last third column , "Deviation", shows the percentage of the difference between the two probabilities for each row relative to the first probability value. The analysis of the column determines that 62% of all deviations between the two experimental probabilities data sets are less then 1%, and 92% are less than 10%. Similar results were obtained from other experiments with different Wiener noise amplitudes.

Data for the Table 3.1 is obtained by computing the probability histogram. In order to build a probability histogram, the total passage time interval should be broken into equal subintervals. A reasonable choice for the number of the subintervals is found empirically to be 25. It is not a magic number. According to the theoretical background given in Chapter 2 it is known that the histogram is supposed to have an exponential tail. Thus, if the total passage time is broken into only few subintervals then it will look like the histogram in the upper half of Figure 3.18. If it is broken into too many intervals then it is discretized to the point where it is

Passage Time Interval	*Probability*		*Deviation, %*
0.0466	0.16857	0.16803	0.32
0.09320	0.14168	0.14254	0.6
0.1398	0.11659	0.11584	0.6
0.1864	0.09813	0.09837	0.2
0.233	0.08167	0.08171	0.05
0.2796	0.06567	0.07145	8.8
0.3262	0.05579	0.05541	0.07
0.3728	0.04616	0.04637	0.45
0.4194	0.03873	0.02835	26.8
0.466	0.03171	0.03172	0.03
0.5126	0.02689	0.02589	3.7
0.5592	0.02107	0.02228	5.7
0.6058	0.01786	0.01807	1.1
0.6524	0.01545	0.01546	0.06
0.699	0.01284	0.01205	6.1
0.7456	0.01003	0.01024	2.1
0.7922	0.008429	0.008432	0.036
0.8388	0.00742	0.00743	0.13

Table 3.1: Stability of the probability distribution for two Duffing ODE box experiments

degree of accuracy. So the output of the experiment should be a graph comparing the set of unstable eigenvalues obtained by using the full set of experimental probability data versus a data set selected automatically by the adaptive least–square fitting algorithm.

3.7 Results

The first experiment with the Duffing model with Wiener noise applied was carried out with $\beta = 0.498, \delta = 0.4, x_0 = 0.2, \dot{x}_0 = 0.3, \varepsilon = 0.25$. These values were chosen based on calculations made by Stone [1] using a Melnikov perturbation method to guarantee that the system has a heteroclinic trajectory. Table 3.1 shows the resulting probabilities for two experiments using different seeds for the randomly generated data. The first column of Table 3.1, "Passage Time Interval", shows the values of the right border of each histogram's subinterval. The first interval starts with zero and goes to 0.0466 time units. The second column, "Probability", is divided into two subcolumns. The first subcolumn displays the calculated probability for the first experiment. The second subcolumn displays the calculated probability for the second experiment. And finally, the last third column , "Deviation", shows the percentage of the difference between the two probabilities for each row relative to the first probability value. The analysis of the column determines that 62% of all deviations between the two experimental probabilities data sets are less then 1%, and 92% are less than 10%. Similar results were obtained from other experiments with different Wiener noise amplitudes.

Data for the Table 3.1 is obtained by computing the probability histogram. In order to build a probability histogram, the total passage time interval should be broken into equal subintervals. A reasonable choice for the number of the subintervals is found empirically to be 25. It is not a magic number. According to the theoretical background given in Chapter 2 it is known that the histogram is supposed to have an exponential tail. Thus, if the total passage time is broken into only few subintervals then it will look like the histogram in the upper half of Figure 3.18. If it is broken into too many intervals then it is discretized to the point where it is

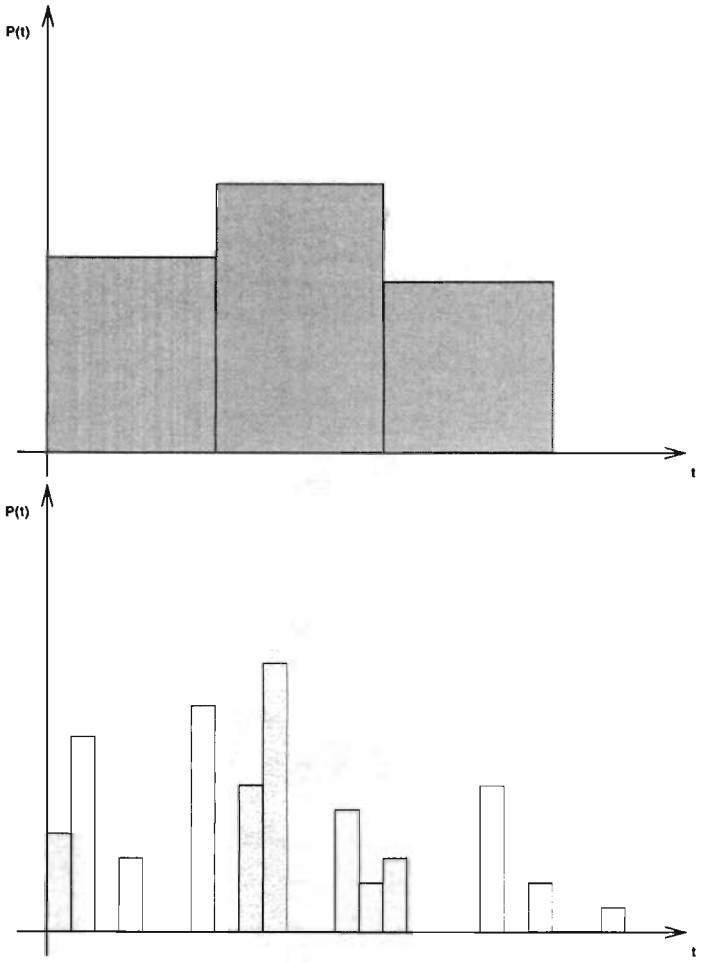

Figure 3.18: Typical passage time histograms discretized with very few and too many subintervals.

impossible to find any pattern as it is shown in the lower half of Figure 3.18.

Table 3.2 shows how increasing the amplitude of Wiener noise affects the calculated unstable eigenvalue λ_u. The experiment was carried out using the Ornstein–Uhlenbeck system box with uniform and Gaussian distributions of the initial value y_0 with which the trajectory enters the box. The table has five columns. The first column "Wiener Noise Amplitude" shows the values of the noise amplitude starting from zero, which means absence of noise, and incremented by 0.1 until it reaches 1.2. The second and third columns represent experimentally obtained unstable eigenvalue λ_u for uniform and Gaussian distributions respectively. The fourth and fifth columns show the relative errors in percentage of variation of the experimentally obtained unstable eigenvalues from the theoretical expectation. If an assumption is made that 10% of the variation is acceptable then up to 0.4% of the Wiener noise amplitude can be applied.

When increasing the amplitude of Wiener noise the accuracy of the resulting eigenvalue changes. In order to demonstrate that, along with showing the effect of increasing the amplitude an experiment was carried out. The experiment was carried out for the uniform distribution of the initial value y_0 with which the trajectory enters the box. The results are quite similar to those of the previous experiment with the Gaussian distribution.

The other parameter that affects the accuracy of the resulting eigenvalue is the box size. An experiment was carried on in order to investigate this effect. Since the Ornstein–Uhlenbeck system is linear, the box size does not influence the result. Table 3.3 shows how the increasing size of the box affects the calculated unstable eigenvalue λ_u for the Duffing's equation. The box side size is a parameter that is chosen arbitrarily. The question arises about choosing this size in order for the computation to be exact. Since the trajectory is nonlinear the box must include only the linear part of the trajectory. When Duffing's equation parameters change, the shape of the trajectory changes too. For the same box side size some percentage of the nonlinear part of the trajectory may be introduced even if it did not happen in the previous computation with different parameters. It is important to see whether it affects the unstable eigenvalue significantly or not. The first column in Table 3.3 shows different values of the box side size. The second column

Wiener Noise Amplitude, ε	λ_u *Uniform*	λ_u *Gaussian*	*Uniform Error* γ, %	*Gaussian Error* γ, %
0.0	1.9897	1.9897	0.519	0.52
0.1	2.05	1.967	2.25	1.52
0.2	1.91	1.928	5.23	3.62
0.3	2.097	2.105	4.74	5.27
0.4	2.164	2.091	8.22	4.55
0.5	1.7	2.363	15.43	18.14
0.6	2.404	1.572	20.19	21.39
0.7	1.56	2.321	21.96	15.62
0.8	2.463	1.638	23.13	17.83
0.9	2.77	2.956	38.31	48.11
1.0	1.218	0.976	39.12	53.61
1.1	3.34	3.261	67.23	63.33
1.2	3.647	3.477	82.37	72.45

Table 3.2: Relative errors of the unstable eigenvalue from noise for Uniform and Gaussian distributions of the initial values in Ornstein–Uhlenbeck box experiment for theoretical $\lambda_u = 2.0$

Boxside Size	$\dot{x}$ Climax	ρ	*Ratio of Discarded Points*	μ %
0.076	1.52	0.05	0.06	14.3
0.152	1.52	0.1	0.33	5.7
0.304	1.52	0.2	0.25	1.6
0.456	1.52	0.3	0.33	6.8
0.608	1.52	0.4	0.42	12.1
0.76	1.52	0.5	0.48	20.7
0.912	1.52	0.6	0.5	33.3
1.064	1.52	0.7	0.5	57.4
1.216	1.52	0.8	0.5	76.9

Table 3.3: Relative error of the unstable eigenvalue for Duffing ODE box experiment

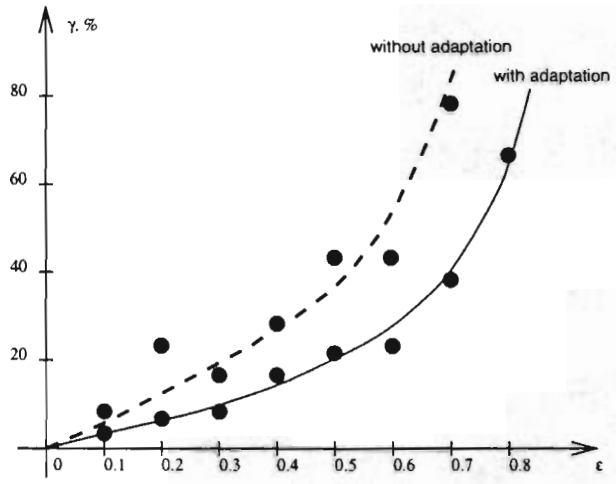

Figure 3.19: Dependency of relative error of the unstable eigenvalue on ε with adaptive least–square fitting algorithm.

shows *climax* value along $\dot{x}$ coordinate. The third column introduces the ratio ρ of these two parameters. This parameter is used to show implicitly the relative size of the side of the box to the *climax*. The fourth column "Ratio of Discarded Points" characterizes the effectiveness of the adaptive least–square fitting algorithm. It indicates how many points were discarded while applying the compression technique that is the essential part of the algorithm. The last, fifth column shows the relative error in the percentage of variation of the experimentally obtained unstable eigenvalue from the theoretical expectation. Table 3.4 has summary information on how the increasing amplitude of Wiener noise and the adaptive least–square fitting algorithm may affect the unstable eigenvalue. The graphical representation of the data given in the Table 3.4 is shown in Figure 3.19.

The graph of the dependency of the average error of discarded points when using the adaptive least–square fitting algorithm on ρ is shown in Figure 3.20. The graph shows nonlinear dependency curve with a minimum at $\rho = 0.2$.

The graph of the dependency of relative error of the unstable eigenvalue from ε in the

Wiener Noise Amplitude, ε	*Climax*	λ_u *theoretical*	λ_u *calculated*	*Error with adaptation* γ_a, %	*Error without adaptation* γ_b, %
0	1.38	2.0	1.9897	0.52	0.52
0.1	1.43	2.0	1.967	1.52	5.3
0.2	1.56	2.0	1.928	3.62	20.7
0.3	1.63	2.0	2.105	5.27	18.6
0.4	1.67	2.0	2.091	4.55	16.1
0.5	1.75	2.0	2.363	18.14	26.3
0.6	1.76	2.0	1.572	21.39	42.3
0.7	1.82	2.0	2.321	15.62	43.5
0.8	1.91	2.0	1.638	17.83	42.4
0.9	1.94	2.0	2.956	48.11	77.6
1.0	2.06	2.0	0.976	53.61	82.3
1.1	2.09	2.0	3.261	63.33	85.4
1.2	2.15	2.0	3.477	72.45	91.7

Table 3.4: Relative error of the unstable eigenvalue with and without adaptive part of least–square fitting algorithm

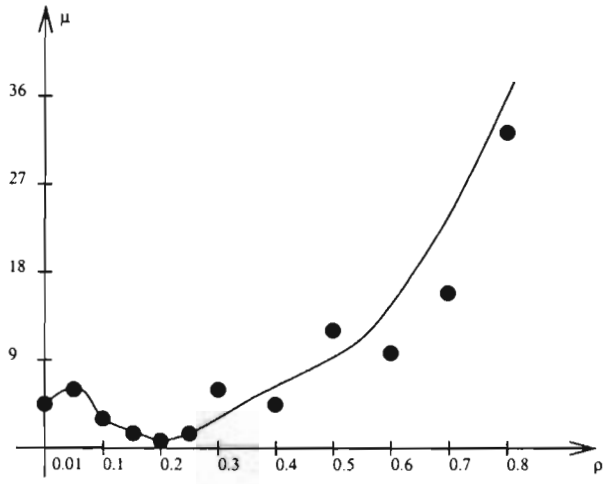

Figure 3.20: Dependency of relative average error of discarded points when using adaptive least–square fitting algorithm on ρ.

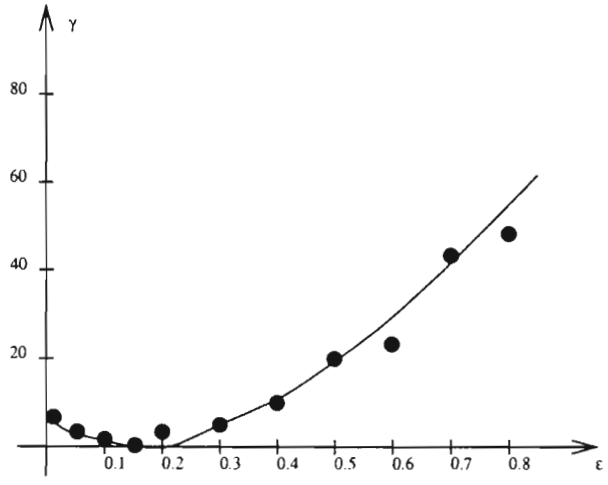

Figure 3.21: Dependency of relative error of the unstable eigenvalue from ε in the Duffing ODE box experiment.

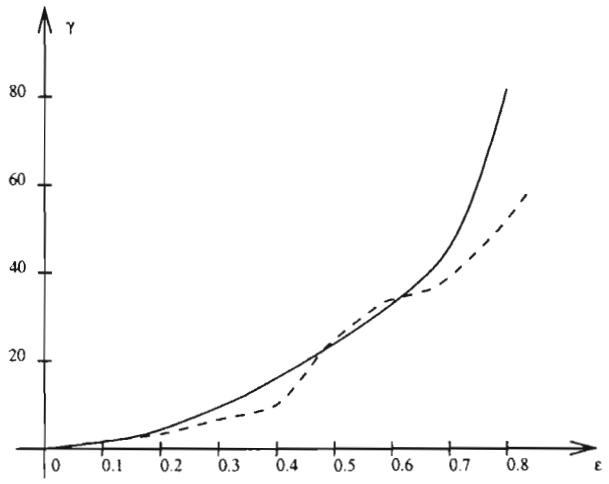

Figure 3.22: Dependency of relative error of the unstable eigenvalue from ε in the Gaussian and Uniform Ornstein–Uhlenbeck box experiments.

Duffing box experiment y_0 is shown in Figure 3.21. The graph shows nonlinear increase of the unstable eigenvalue error when increasing the amplitude of Wiener noise added to the system.

The graphical representations of the dependency of relative error of the unstable eigenvalue on ε in the Ornstein–Uhlenbeck box experiment for Gaussian distribution of the initial value y_0 is shown in Figure 3.22 with a solid curve. The graph shows nonlinear increase of the unstable eigenvalue error when increasing the amplitude of Wiener noise added to the system. The graphical representations of the dependency of relative error of the unstable eigenvalue on ε in the uniform Ornstein–Uhlenbeck box experiment is shown in Figure 3.22 with dashed curve. The graph shows nonlinear increase of the unstable eigenvalue error when increasing the amplitude of Wiener noise added to the system. The initial value y_0 is distributed uniformily.

The experimental results for the numerical experiment with Duffing system are shown in Figure 3.23. Time units are represented by axis X and the distributions of the probabilities are represented by axis Y. After running the **HeteroCon** program over the data accumulated during the computational experiment with the system, the distributions of the probabilities are

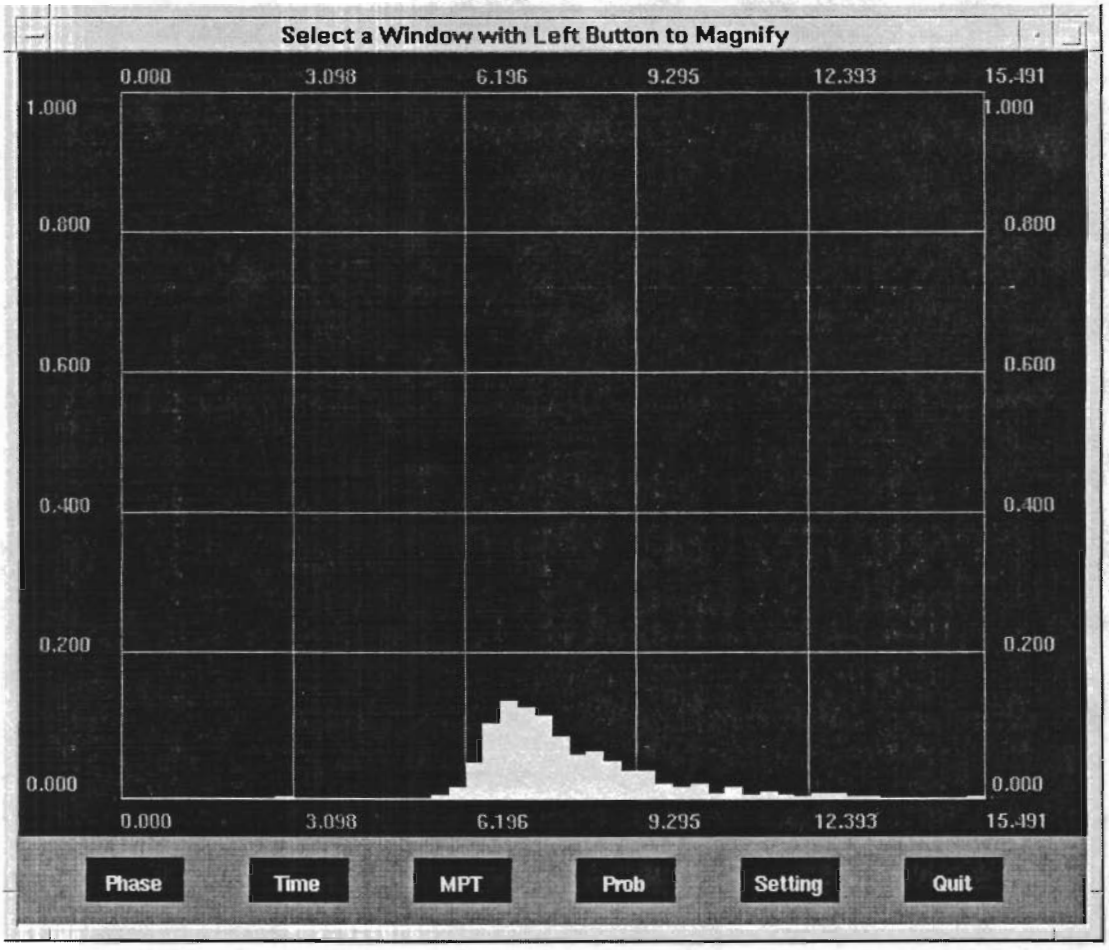

Figure 3.23: The histogram of the probabilities for the Duffing box experiment.

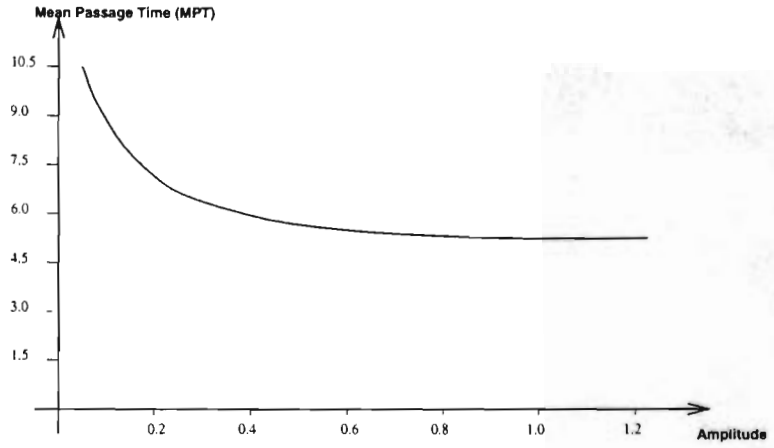

Figure 3.24: Graph of experimentally obtained dependancy of MPT from Wiener noise amplitude for the Duffing box experiment.

shown with white rectangles. They form a curve with shape close to a Gaussian distribution. The curve consists of two exponential curves with different slope coefficients. The presence of noise blurs the shape, but nevertheless the strong exponential components are clearly seen.

The graph of the dependancy of the mean passage time MPT value on the amplitude of Wiener noise is shown in Figure 3.24. The purpose of this graph is for comparison with the results shown in Stone [1]. The results show that upon increasing the level of the Wiener noise amplitude, the MPT exponentially decreases until it reaches some saturation point. The value of saturation is about 6 time units in this case. It confirms the existence of the presence of some optimal path that the system chooses when applying an external noise process.

The experimental results for a numerical experiment with the Ornstein–Uhlenbeck system are shown in Figure 3.25. Time units are represented by axis X and the distributions of the probabilities are represented by axis Y. After running the **HeteroCon** program over the data accumulated during the computational experiment, the distributions of the probabilities are shown with white rectangles. The theoretically calculated distributions are shown with dark grey rectangles superposed over the white ones. Even without using any measuring tools it is

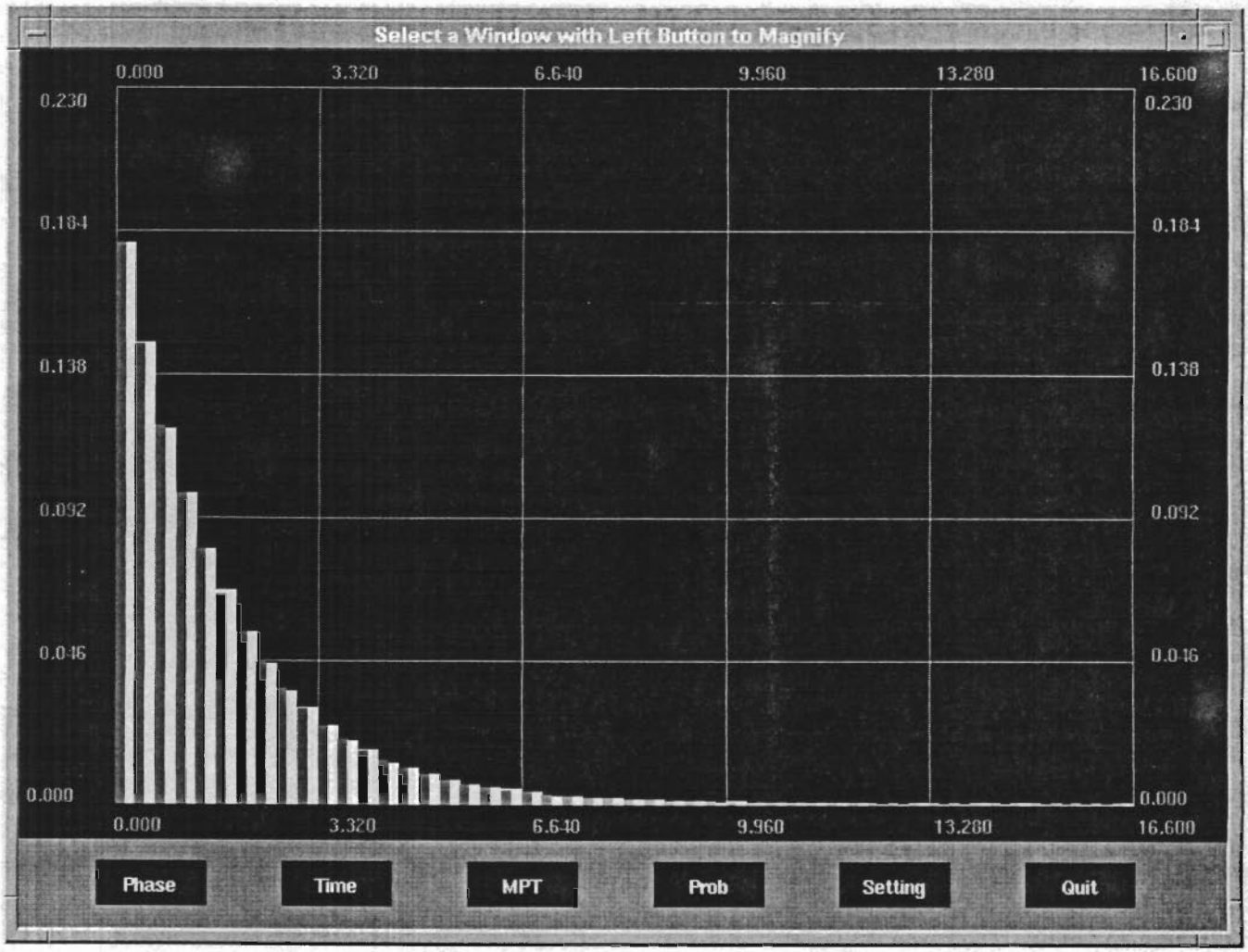

Figure 3.25: The experimental histogram of the probabilities for Ornstein–Uhlenbeck system.

easy to see there is a little discrepancy between the experimental and theoretical data.

3.8 Summary of Numerical Experiment Results

In a natural physical flame system, a small additive noise can occur naturally due to small disturbances in the fuel supply and the chemical balance. In this case it is very important to give estimates of how this noise can affect the descriptive parameters. The numerical experiments show that small additive noise in the range from 0 to approximately 25 percent of y_{max} does not affect the unstable eigenvalue significantly. Let's take a look at the relative errors of the unstable eigenvalue in Table 3.2. Although quite similar, the errors are smaller in most of the cases for Gaussian distribution of the initial values. One possible explanation is that having properties of Wiener noise such as zero mean variance and limited amplitude, the deviation of the initial value is quite predictable. If the initial value is close to any of the box side ends then it can be moved by the noise to the center of the box side. If the initial value is close to the center of the box side then it can be moved to the one of the ends. When one type of shift of the initial value prevails, it is more correctable in case of the Gaussian distribution since it already has an exponential multiplicator $e^{-y_0^2+\mu}$. By using an adaptive least square fitting algorithm, it is possible to sift out the unwanted results and recover the genuine picture of the passage times distribution. Such small sensitivity to external noise can be expained after careful study of the Ornstein–Uhlenbeck system without noise. The probability distribution has a maximum corresponding to passage times with initial conditions starting in the middle of the box side and laying in its neighborhood $\Delta\dot{x}$. When a dynamic system with heteroclinic behavior is considered the applied Wiener process allows a simulated heteroclinic orbit to stay near the ideal one. The box drawn around a hyperbolic fixed point usually accepts a heteroclinic orbit in the middle of its side with $\pm\Delta\dot{x}$. Wiener noise may move the trajectory away from the neighborhood but it does not happen too often. So the distribution looks the Gaussian as in Figure 3.23. During the experiment it was found out that only the first six or seven points contribute with very high

accuracy to the values of the coefficients.

Chapter 4

Heteroclinic Connection Detection Algorithm

4.1 Description of the Problem

This chapter describes the proposed computer techniques for detection and analysis of heteroclinic connections in sequences of flame images. The combustion flame is a multivariable dynamic system exhibiting extremely complex behavior. The previous sections of this thesis concentrated on relatively simple systems that could be analyzed using two–dimensional phase plane techniques. These approaches are not directly applicable to the flame system if copied directly from the given examples.

Heteroclinic behavior in the flame system is observed experimentally as follows. The system may be self–organized in one of the spatial patterns described in [18]. It evolves rapidly by changing patterns. Finally, it again enters a state of homeostasis where it stays for a relatively long time without significant changes. Then the system jumps into the whirlpool of rapid multiple changes and afterwards moves to some new stable state. It is quite possible that the system may revisit some of the previous stable states. According to the Manneville–Pomeau

theory given in Chapter 1, if it were possible to find a single variable that represents the state of the system, then the behavior could be explained by looking at a one–dimensional map. When the state of the system is located close to the bisectrix of the map coordinate system then, for an outside observer, the system would seem to be frozen. However, when the system gets out of the bisectrix neighborhood, the behavior will change rapidly. The definition of heteroclinic connection was given in Chapter 2. It was shown in Chapter 2 that the system's behavior, as it approaches or leaves a stable state, is exponential. This exponential behavior may serve as an identification method for finding heteroclinic connections in a system's state diagram where the number of states is very large. The main achivement of algorithms and techniques developed in this chapter is that heteroclinic connections in a real physical system can be detected on the fly as the experiment goes. This real–time detection tool can be used by researchers to quickly detect and predict the state of the system. Sections 4.2 and 4.3 explain how to use differential energy to build a graph of the system's state. Sections 4.4 and 4.6 describe a general algorithm comprising a number of techniques to identify heteroclinic connections. Section 4.5 gives an overview of the neural computation methodology. Section 4.7 describes the Madaline neural network that is used to identify exponential behavior. Section 4.8 describes the essential components of the software created to implement the algorithm.

4.2 Differential Energy

A full description of the state of the flame system includes geometrical quantities characterizing the spatial structure into which the flame self–organizes; illuminational parameters describing the brightness of parts of the flame; signal–to–noise ratio and many more. The state matrix comprising these parameters is multidimensional. The analysis of evolution of this matrix would require enormous computing resources. Indeed, it would require not only tracking each component of the matrix, but also evaluating the correlation of the changes of the component with changes of other components. Because of the complexity of the system, one looks for

global quantifiers to characterize the behavior. Let's introduce a parameter called differential energy. Imagine that every pixel of a flame movie frame represents a particle. The energy of each pixel is defined by its value. *The square of the difference of the values between two corresponding pixels in consecutive frames is the differential energy of the pixel. The normalized sum of differential energies of all pixels constituting two consecutive frames is called differential energy.* It is described by:

$$E(t) = \frac{1}{N^2}\Sigma_{ij}^{N}(m_{ij}(t) - m_{ij}(t-1))^2 \tag{4.1}$$

where $m(t)$ is a frame t from a flame movie, i and j are the coordinates of a pixel of the movie frame, t is discrete time labelling the frame number, N is the total number of frames used to calculate differential energy.

A sample graph of differential energy versus frame number is shown in Figure 4.1. In general, it looks like a stochastic time series. Many factors contribute to the stochastic look. If two images are exactly alike, then the differential energy is equal to zero, since the corresponding pixels in the two images have the same values. If one image contains only pixels having zero value, and the second image contains pixels having a maximum value, then the resulting differential energy of the images is the maximum. An infinitesimal change in the value of a pixel in one image leads to a similar infinitesimal change in the differential energy. Taking these assumptions as basis for image differential energy calculus, one can assume that this crude approach can allow a researcher to provide a qualitative description of image dynamics. A sequence of images may be substituted by a time series of differential energies plotted vs flame numbers. The latter serve as discrete time.

4.3 Exponential Patterns in Differential Energy Graphs

Chapter 3 mentioned that when a dynamic system moves away from a saddle point, its solution carries a very strong exponential component associated with a positive eigenvalue of

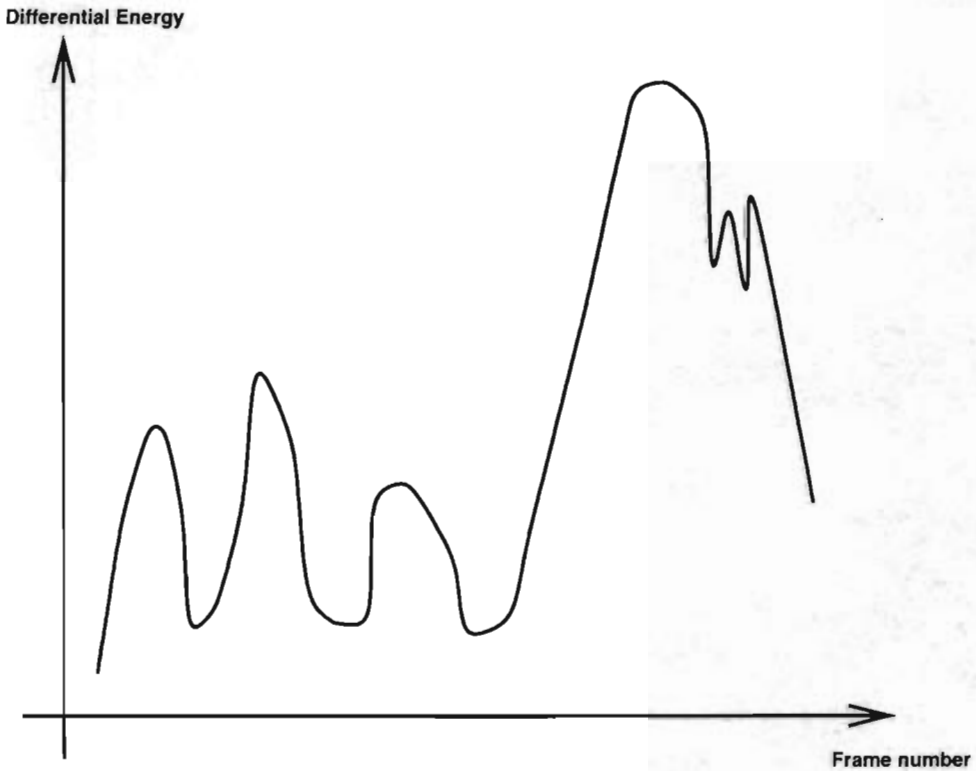

Figure 4.1: A sample graph of differential energy.

the linearization around the fixed point. When the system is in the transition away from a saddle point, the corresponding graph of differential energy shows an exponential growth. Let's say the graph is composed of many geometrical patterns: linear, polynomial, sinusoidal, etc. True exponential patterns that correspond to passage close to a saddle occur during a small period of the lifespan of a real dynamic system. Detecting exponential patterns is important, because it provides information for computing the probability distribution of times that a system of passage times used to characterize exponential patterns. In the past, the basic methodology for analyzing heteroclinic connections was to use a domain expert or analyst to identify exponential patterns. The expert analyzes the dynamic system in order to detect all points of interest. He posesses special knowledge about the system's behavior that allows him to identify all the occuring events. This methodology has several problems: It is very time consuming. Having an analyst look through an entire data set in which the interesting pattern occurs only once in a blue moon, is very nonrewarding. When the system has noise or many abnormal patterns, the analysis task is even more difficult. Real dynamic systems have many variables, all of them are changing with time. It can be very troublesome for even an experienced analyst to track a pattern that undergoes qualitative changes over a long period of time. Reducing a multidimensional system to a one–dimensional time series system is the first step in alleviating this cumbersome task. A sample graph of differential energy with an exponential pattern in it is shown in Figure 4.2. The solid line depicts the graph, and the dashed line shows an approximation of the exponential behavior.

4.4 Recognition of Exponential Patterns

In order to build the probability distribution graph of passage times between heteroclinic connections, exponential behavior should be identified first. The source data set is a set of movie frames. Each movie frame shows spatio–temporal patterns, and there is no accepted system for classifying these patterns. A great deal of general pattern classification in time series

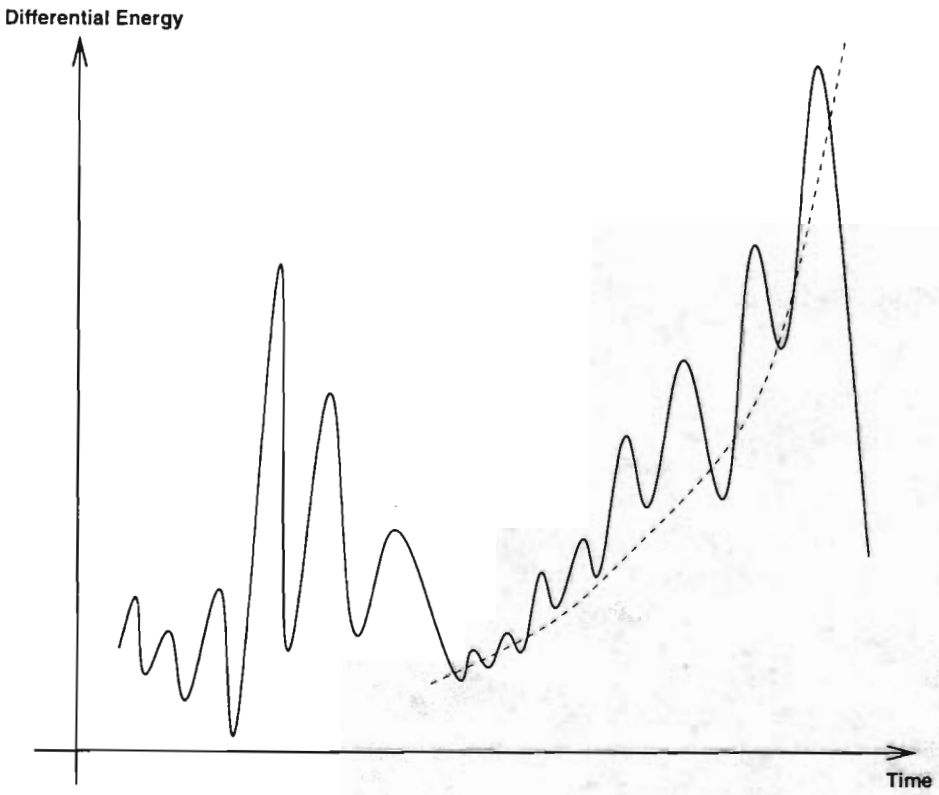

Figure 4.2: A sample graph of a differential energy with an exponential pattern.

was done in [19]. The nature of the data to be analyzed in this thesis falls under the category of signal processing. The main idea is to remove or significantly reduce unwanted components from the input signal. The information that is of interest to a researcher is encoded in the differential energy time series in patterns of exponential growth. In order to identify these patterns the signal noise and other noninteresting patterns should be filtered out. A straightforward approach would dictate the use of a combination of lowpass, highpass, bandstop, and bandpass filters. Fourier analysis would be performed over the time series to analyze the frequency–domain effects in the time–series signal.

Because of the inherently noisy nature of a differential energy graph obtained from flame movie data, it is time consuming and complicated to identify exponential patterns. Thorough classification of flame patterns was done in [18]. However, it is difficult to obtain such properties of flame spatio–temporal patterns that can be used efficiently for building characterization and comparison algorithms. Dynamic system methods are inadequate to solve this problem because they are intended to highlight hidden characteristics of dynamic systems. These methods can hardly be used to determine relative changes in spatial structures. Alternative tools such as Fourier transformations and correlation functions provide insight for systems that are ordered in space and time, but the inadequacy of the power spectrum even for analyzing time series in spatially homogeneous systems became widely recognized more than a decade ago. Researchers [19] realized that the same broadband power spectrum could arise from a low–dimensional dynamic system or from a very high–dimensional system. Multiple limitations of Fourier spectra became an issue in the early 1980s, and these problems caused experimenters to apply dynamic system methods for analyzing attractors construed from time series. These techniques, to some extent are still used to analyze temporal behavior of dynamic systems. Other techniques including wavelet transformations [20] and proper orthogonal decompositions [19] provide some quantitative characterization of patterns but fail to distinguish different types of dynamics. A need for new techniques is acknowledged, and the importance of their application to various dynamic systems is clearly recognized [19]. Suppose that there exists a transforma-

tion operation R, and it is applied to the digitized approximation of the input signal $x(n)$ for each input sample. The output of this operation is a value $y(n)$ where n is a discrete timestep corresponding to a frame number. This transformation operator R is in fact a digital filter. The benefit of this formulation is that once the system response to the sample function is known, the system output for any input can be calculated using repetitive mathematical operations over the input data domain. The domain can be thought of as a landscape. The operator is a window sliding over this landscape to find a scene of interest.

4.5 Neural Computation

Many papers and books have been published on the subject of neural computation. The reason for much of excitement about neural networks is their ability to generalize to new situations [21]. After being trained on a number of examples of a relationship, they can often induce a complete relationship that interpolates and extrapolates from the examples in a sensible way. Neural networks commonly make very useful generalizations that would be judged sensible in human terms. A definition of a neural network is given in [22]. An artificial neural network is an information–processing system that has certain performance characteristics in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on assumptions that

1. Information processing occurs at many simple elements called neurons.
2. Signals are passed between neurons over connection links.
3. Each conection link has an associated weight, which, in a typical neural net, multiplies the signal transmitted.
4. Each neuron applies an activation function to its net input to determine its output signal.

A neural network is characterized by its pattern of connections between the neurons or architecture, its method to determine the weights on the connections called its training, or learning algorithm, and its activation function. A thorough description of neural networks is given in [22].

4.6 General Algorithm

Heteroclinic connections can be identified by finding all intervals of the graph exhibiting exponential behavior. However, this task is complicated by the presence of significant noise. Straightforward computational procedures such as numerical differentiation or logarithmic conversion of the graph followed by application of the least–square method are CPU intensive. In addition to that, the presence of noise requires additional statistical analysis, otherwise the standard methods may not work. The key to the solution of the problem is that a researcher can identify exponential behavior just by looking at the graph for a relatively short time. Obviously, the human brain has a quality that standard computational techniques can not offer, suggesting the use of the artificial neural network technology as a viable solution to the problem.

4.6.1 Conversion of a Differential Energy Graph

Most neural networks operate on binary or bipolar input. Binary input is represented as an array of 1 and 0 values, and bipolar as 1 and -1 values. At the same time the graph of differential energy is an array of floating point numbers. Thus, a conversion mechanism should be offered to digitize the array, and represent it in binary or bipolar form. Let's introduce a notion of a digitizing window. This window is divided into checks. The size of each check is Δx along axis x and Δy along axis y. Let us superpose it with a graph of differential energy as shown in Figure 4.1. The x axis of the graph of differential energy represents a frame number and takes only discrete value. Axis y represents differential energy that takes continuous values. Thus,

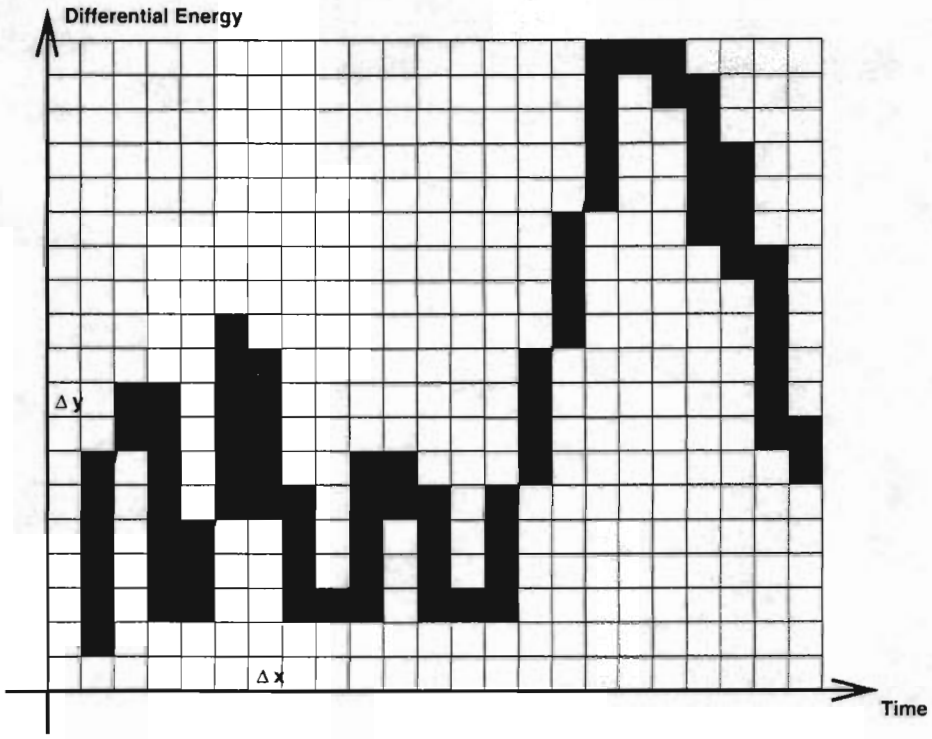

Figure 4.3: An example of digitized graph of differential energy.

if $\Delta x \rightarrow 1$ and $\Delta y \rightarrow 0$ the discrete representation will transform into the original graph. It is important to take values of Δx and Δy so that the uniqueness of the original graph will be preserved. At the same time they should not be very small, otherwise the time spent on computational process will increase indefinitely. The output of this procedure is a two–dimensional matrix. The following example uses the graph shown in Figure 4.1. A rectangle template made out of transparent boxes is superposed over the graph. The following rule is used to color a box. If a point of the graph falls into a box, then the box is colored. The resulting graph is shown in Figure 4.3.

In general, a differential energy graph may contain a very large number of exponential patterns. Each pattern has different geometrical characteristics. They vary in height, width,

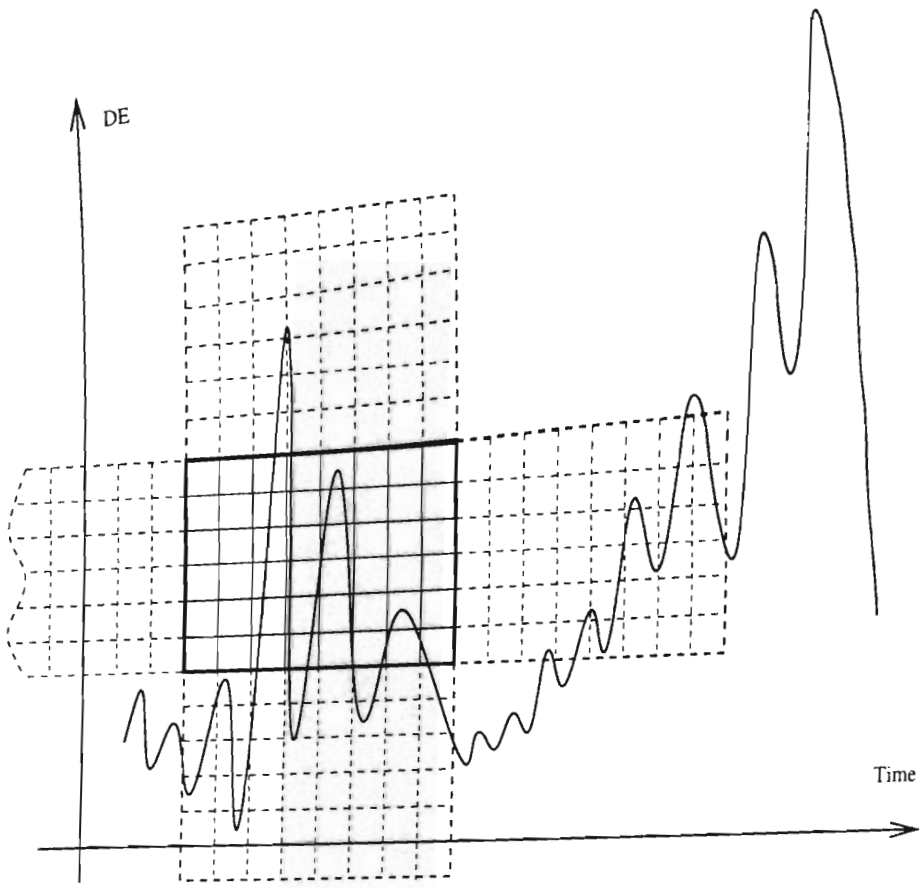

Figure 4.4: Applying sliding digitizing window to a differential energy graph.

signal–to–noise ratio, and other parameters. In order to identify all patterns, the size of Δx and Δy should be selected for each group of patterns that has similar geometrical properties. In the previous section it was mentioned that a sliding window moves across the graph's landscape to find the scene of interest. This window is shown in Figure 4.4 as a solid rectangle positioned over a differential energy graph. This rectangle is divided into checks as shown in Figure 4.3. The rectangular windows depicted by dashed lines represent the same window moved toward all directions to digitize the graph.

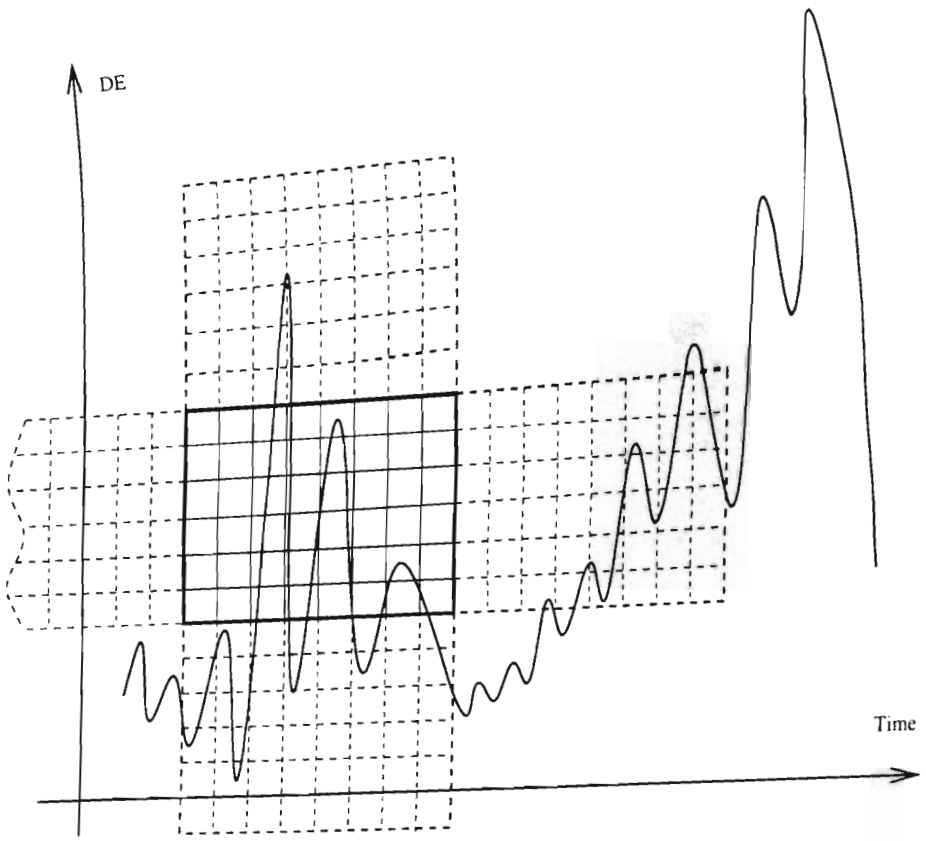

Figure 4.4: Applying sliding digitizing window to a differential energy graph.

signal–to–noise ratio, and other parameters. In order to identify all patterns, the size of Δx and Δy should be selected for each group of patterns that has similar geometrical properties. In the previous section it was mentioned that a sliding window moves across the graph's landscape to find the scene of interest. This window is shown in Figure 4.4 as a solid rectangle positioned over a differential energy graph. This rectangle is divided into checks as shown in Figure 4.3. The rectangular windows depicted by dashed lines represent the same window moved toward all directions to digitize the graph.

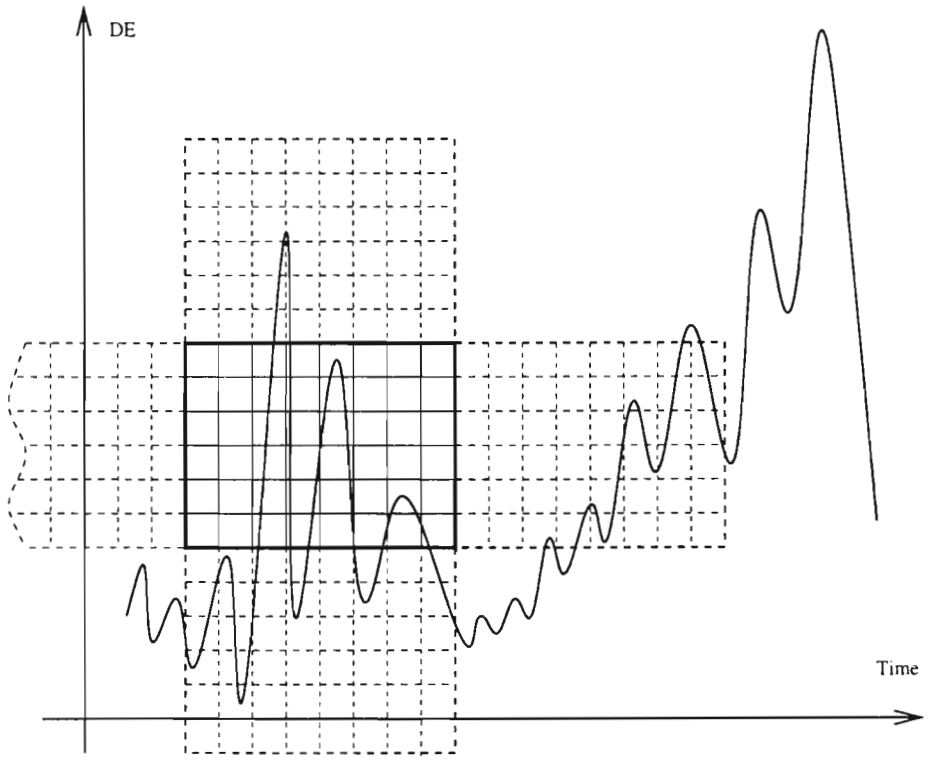

Figure 4.4: Applying sliding digitizing window to a differential energy graph.

signal–to–noise ratio, and other parameters. In order to identify all patterns, the size of Δx and Δy should be selected for each group of patterns that has similar geometrical properties. In the previous section it was mentioned that a sliding window moves across the graph's landscape to find the scene of interest. This window is shown in Figure 4.4 as a solid rectangle positioned over a differential energy graph. This rectangle is divided into checks as shown in Figure 4.3. The rectangular windows depicted by dashed lines represent the same window moved toward all directions to digitize the graph.

4.6.2 General Algorithm

The sequence of steps for the general algorithm to indentify exponential patterns is the following.

Step 1. Initialize Δx and Δy. Define the start value x_{start} for the differential energy graph. Define a submatrix size.

Step 2. Digitize the differential energy graph.

Step 3. For each submatrix of the digitized differential energy graph, until the end of the graph is reached, do

Step 4. Build an input array for a neural network

Step 5. Compute the output of the neural network.

Step 6. Record the submatrix coordinate if an exponential curve has been detected

Step 7. Define a new coordinate for the submatrix.

After a differential energy graph is digitized and the two–dimensional matrix is produced, a submatrix is chosen. The submatrix has the same number of rows as the main matrix. The selection starts with the first column and covers the predefined number of columns. The chosen submatrix is consequently transformed into an one–dimensional array, and fed into a neural network. The output produced by the neural network shows whether an exponential curve has been detected in the input. If it has been found, then notification is made about the detection of an exponential curve. The start column is shifted with the number of columns in the submatrix, and the algorithm continues. Finally, when the final column in the data matrix is reached, the algorithm stops.

4.6.3 Exponential Connection Verification Algorithm

The detection of exponential patterns does not necessarily imply a trajectory passage close to a saddle. The following sequence of steps shows a verification algorithm that verifies whether a detected pattern is really a passage close to saddle by comparing numbers of cells in both inner and outer rings formed by cellular spatio–temporal organization of flame. In order to do that additional information is needed such as number of cells and their structure for cellular flame spatial organization.

Step 1. Generate a list of frame numbers showing detected exponential fall–off patterns followed by detected exponential growth patterns.

Step 2. Supply a list of frame numbers, where each number is followed by a number of cells in the inner ring and a number of cells in the outer ring.

Step 3. For each sequence of exponential fall–off and growth pattern frame numbers

Step 4. Interpolate the number of cells in both rings from the cell list.

Step 5. Check to see whether the number of cells on both rings changed when between fall–off and growth patterns.

Step 6. If the numbers did not change then record the patterns as passages.

Step 7. If the numbers changed then discard the patterns.

4.7 Madaline Neural Network Description

Artificial neural networks are widely used in areas where it is necessary to qualify a system's behavior in the way biological neural networks do. Artificial neural networks are used in this thesis because of the complexity of the dynamical system considered. In order to make a judgement about whether the system changed its behavior, a lot of experimental data has to be collected, processed and numerical characteristics obtained. These characteristics should later

be compared with predefined threshold data. As a result of the comparison a conclusion will be made about whether the dynamical system belongs to one or the other behavioral category. A human decision–making process is based based purely on the observation of the system. One does not have to measure anything. It is quite the contrary for a deterministic algorithm. It is very hard to forsee all varieties when developing such an algorithm. Assuming that such a program has been developed, it is easy to predict that it is cludgy and not robust. In cases like that, neural networks offer the help needed.

A very simple processing element of neural networks is shown in Figure 4.5. In [21] it is thoroughly explained how the simplest neural networks function. To recapitulate briefly, the processing element performs a sum–of–product calculation using the input and weight vectors, and applies an output function to get a single output value. Using the notation from the Figure 4.5, a formula for the output value y is $y = w_0 + \Sigma_{j=1}^{n} w_j x_j$. Adaptive neural networks have a well–defined procedure for modifying the weights in order to allow the device to give the correct output value for the given input. A good example of such procedure would be the delta rule defined below.

When input vectors provide a steady pattern, it is possible to use a **Madaline neural network**. Why is Madaline neural network chosen for this research? To answer this question it is necessary to consider the class of neural networks that Madaline belongs to. The class is called *pattern classification neural networks*. As indicated by the name, the networks that constitute this class provide some mechanism to distinguish whether a specific pattern belongs to some predefined class. The theoretical roots for this class of neural networks lie in the **delta rule**. The main idea of the delta rule is in changing the weights of a neural network during training so that the difference between the input and output units becomes minimal. A derivation of the delta rule is given in [22]. The formula for adjusting the Ith weight for each pattern is

$$\Delta w_I = \alpha(t - y_{in})x_I \tag{4.2}$$

where α is a learning rate, t is target output, y_{in} is the net input to output unit Y, w_I is the weight

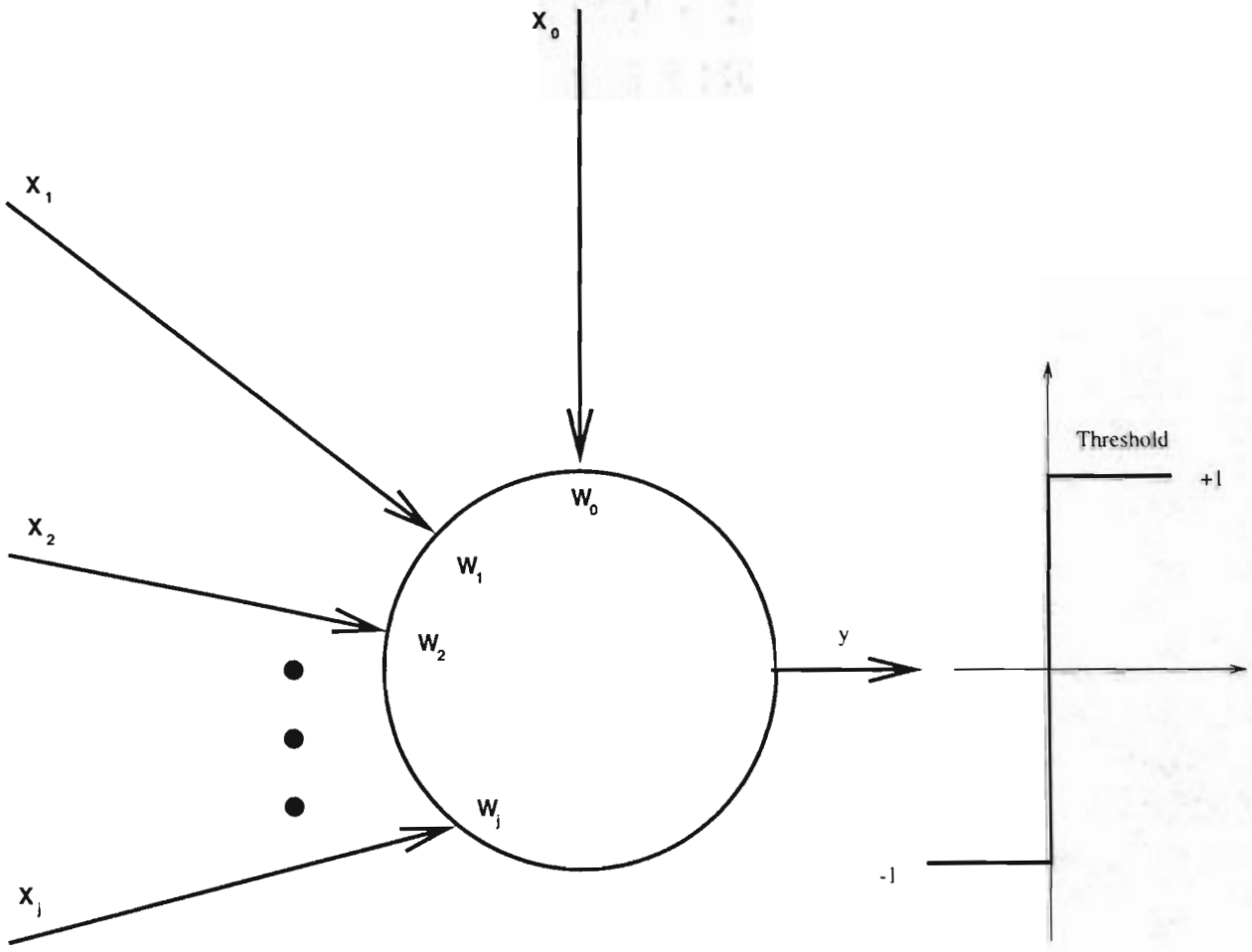

Figure 4.5: Adaptive linear combiner as the simplest processing element of Madaline neural network.

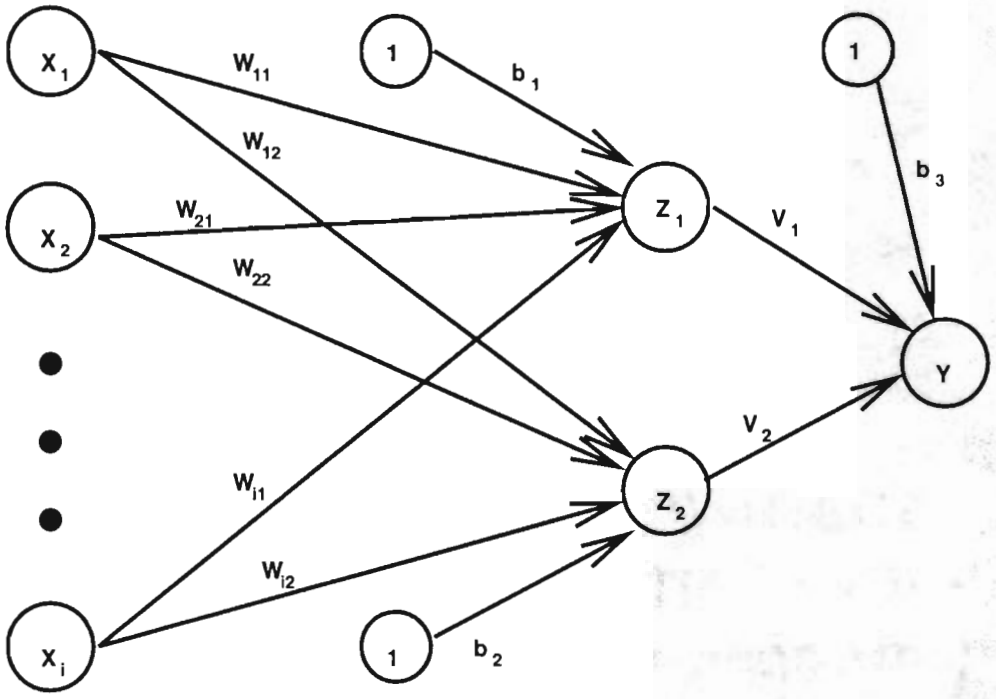

Figure 4.6: Madaline neural network architecture.

of input unit I. A number of papers [23], [24] have been published that generalize the **delta rule** for various cases. For example, training data for larger data sets, or to handle probability distributions. Correspondingly, different neural network architectures were created that employ the **delta rule**. The **Madaline neural network** is one of them. *Madaline* stands for Many ADaptive LInear NEurons. As the name indicates, many neurons are arranged in a multilayer net. Its training is as straightforward as for simpler nets like the *perceptron* or *Adaline*. At the same time, due to the more complex architecture, Madaline gives better precision in recognizing complex patterns. The net tries to update the weights only if an error occured, and in such a way that it is more likely for the net to produce the desired response. The architecture of the network is shown in Figure 4.6. The algorithm is shown below. It is described in more detail in [22].

Input values for this neural network are bipolar, either 1 or -1. The activation function for units Z_1, Z_2, and Y is

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{4.3}$$

Step 1. Initialize weights w_{ij} with small random values. Set learning rate α and biases b_1, b_2, and b_3.

Step 2. While weights do not converge do

Step 3. Set the activation of input units $x_i = s_i$.

Step 4. Compute the net input to each hidden unit: $z\_in_1 = b_1 + x_1w_{11} + x_2w_{21} + \ldots + x_jw_{j1}$, $z\_in_2 = b_2 + x_1w_{12} + x_2w_{22} + \ldots + x_jw_{j2}$.

Step 5. Calculate the output of each hidden unit: $Z_1 = f(Z\_in_1)$, $Z_2 = f(Z\_in_2)$.

Step 6. Calculate the output of net $y\_in = b_3 + Z_1v_1 + Z_2v_2$, $y = f(y\_in)$.

Step 7. Determine error and update weights: if $t = y$, no weight updates are performed. Otherwise: if $t = 1$, then update weights on Z_J, the unit whose net output is closest to 0, $b_J(new) = b_J(old) + \alpha(1 - z\_in_J)$, $w_{iJ}(new) = w_{iJ}(old) + \alpha(1 - z\_in_J)x_i$. if $t = -1$, then update weights on all unit whose net output is positive, $b_J(new) = b_J(old) + \alpha(-1 - z\_in_J)$, $w_{iJ}(new) = w_{iJ}(old) + \alpha(-1 - z\_in_J)x_i$.

4.8 Exponential Pattern Detection Software

The process of identifying heteroclinic patterns is complex and consists of repetitive mathematical calculations. Not only should the neural network be trained in the beginning, but it could also require additional retraining on the fly as an analyst detects that the identification process goes wrong. The software should therefore provide both semi–automatic and automatic detection to allow researchers to either walk through detection process carefully or get results

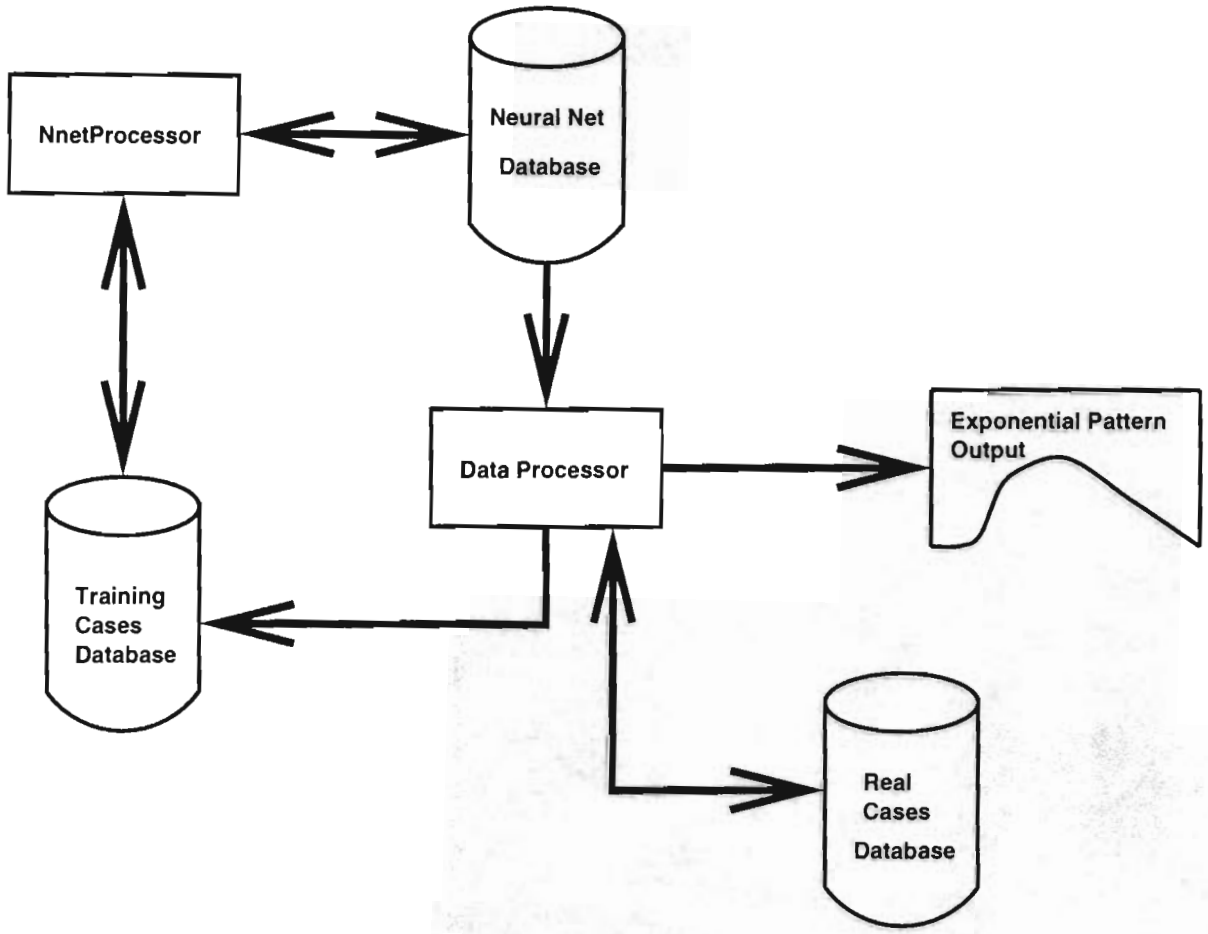

Figure 4.7: Block schema of software developed in this thesis to detect heteroclinic patterns.

as fast as possible. When designing software for detection of exponential patterns, a distributed modular approach was chosen versus a monolitic one. A block schema of the software is shown in Figure 4.7. It consists of two major components that are responsible for the implementation of the exponential pattern detection algorithms developed in this thesis. These components are **NnetProcessor** and **DataProcessor**. In Figure 4.7 they are shown using solid rectangular boxes. The arrows show data flow between software modules.

Figure 4.7 shows three databases: Neural Net Database, Real Cases Database and

Training Cases Database. The Neural Net Database keeps information about neural net input size and weights. This information is used to verify whether input data sets of compatible size are submitted. The Training Cases Database is used to keep both synthesized and real patterns to train the neural network. The Real Cases Database keeps digitized data sets obtained from real dynamic systems. NnetProcessor uses the Neural Net and Training Cases Databases in order to initialize neural network components and to train the network. It also has the capability of creating and adding training patterns to the Training Database as well as manipulating information in the database. DataProcessor uses all three databases. It can only add patterns to the pattern databases and use the database to retrain neural network on the fly. The DataProcessor produces an output file containing a sequential list of detected exponential patterns that is used by the module to build the probability distribution graph of passage times through heteroclinic connections.

4.8.1 Event Logger

The differential energy data analysis software consists of different distributed processing components. Depending on the size of a data set, the task of finding exponential patterns may take a long time. In the middle of an execution cycle a problem may occur that deserves the researcher's attention. Every software component needs to log some information into a database called the Event database. The purpose of such logging is twofold.

An analyst needs to see it to either verify that the algorithms ran correctly or, in case of an error to find out what component caused it. In other words, to analyze and correct the problem. As mentioned already, this software package comprises a number of components that either run in their own process space, or attach to other process spaces. It is much easier for an analyst if all events are put into a single database. It also makes sense not to duplicate the event logger in RAM for every component that uses it. It means that the concurrency problem should be solved for this component. An analyst may also want to control the size of the event database

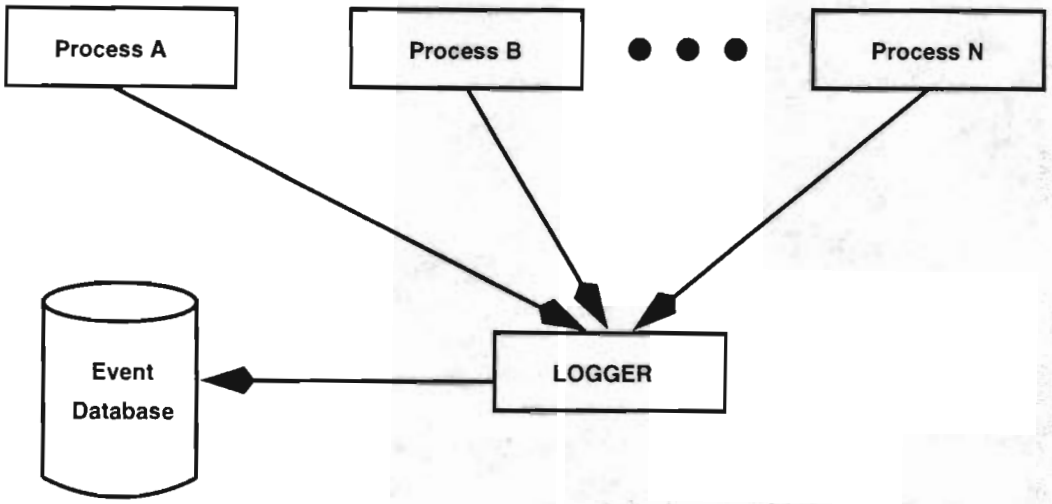

Figure 4.8: Event logger component block schema.

and subsequently the performance of the software package by logging only severe events and getting rid of mundane readings. The block schema of the Event Logger in Figure 4.8 shows a number of processes sending events concurrently to the Logger. The Logger resolves the concurrency, and logs events in the order received. Programmatically, the following C++ statement is used to log an event:

```
LOG.dout( SEVERE, __FILE__, __LINE__ ) << "Text message " <<
 string_value << "; " << integer_value << LOGEND,
```

where SEVERE is the event threshold, __FILE__ is a C macro that is substituted by a file name, __LINE__ is a C macro that is substituted by a line number in the file, LOG is a global Logger object, dout is the class output method, LOGEND is the class' macro that designates the end of the logged event, and << is an overloaded output stream operator. A sample output for this statement would be:

```
12:07:36 Dataprocessor CDataProcessor.cpp line 127:
Text message string value here; 1234567890.
```

4.8.2 Pattern Database Structure

The training pattern database is shown in Figure 4.7 keeps information about real and synthesized patterns used to train the neural network. Both databases consist of a main header and patterns. The main header starts with a security descriptor that occupies 19 bytes. It is a unique text string used to identify the database file. The second control element is common for both databases. It keeps a number of total pattern cases held in the database and occupies 4 bytes. The third element in a training database is a 4 byte integer that keeps the number of rows in the pattern matrix. The fourth element in the training database is a 4 byte integer that keeps the number of columns in the pattern matrix. The layout is shown in Figure 4.9. Right below the layout of the main header, the layout of a data element is shown. The first byte shows whether the element is marked to be deleted from the database. If it is deleted than it is bypassed by the training algorithm, and is physically removed from the database when the compression routine developed in this software package is run. The second byte shows whether the pattern is exponential. Finally, the pattern data is listed. Its size is equal to the product of the number of columns and the number of rows given in the main header.

In case of a real pattern database, the pattern data layout is the same as in the training pattern database. The main header contains a 19 byte security descriptor, a 4 byte integer that holds the number of real pattern cases in the database, and two 4 byte integers holding numbers of rows and columns of the digitizing template. Every pattern data is preceded by two 4 byte floating numbers holding information about the length and height of the digitizing template in real differential energy graph metrics. The next two 4 byte integers hold the numbers of rows and columns in the digitized differential energy graph, followed by the digitized graph data. Its size in bytes is equal to the product of rows and columns numbers.

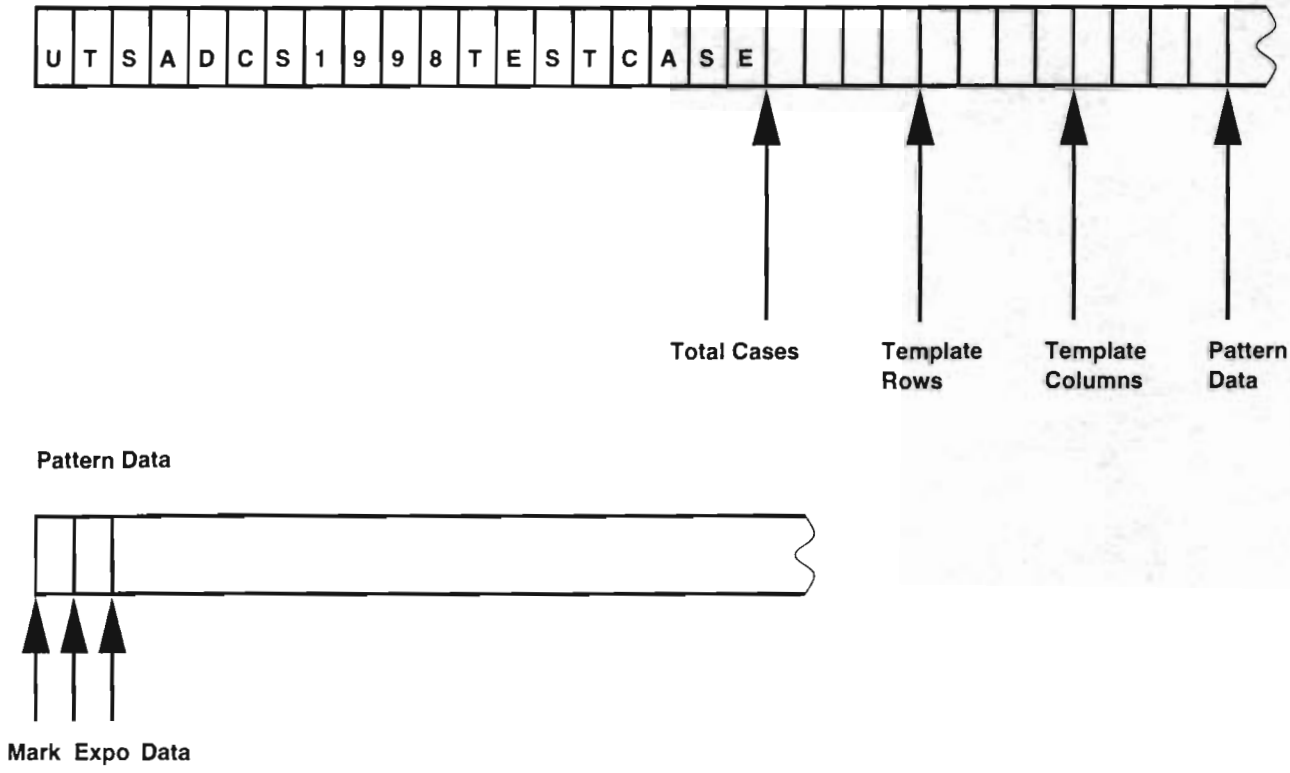

Figure 4.9: Training patterns database layout.

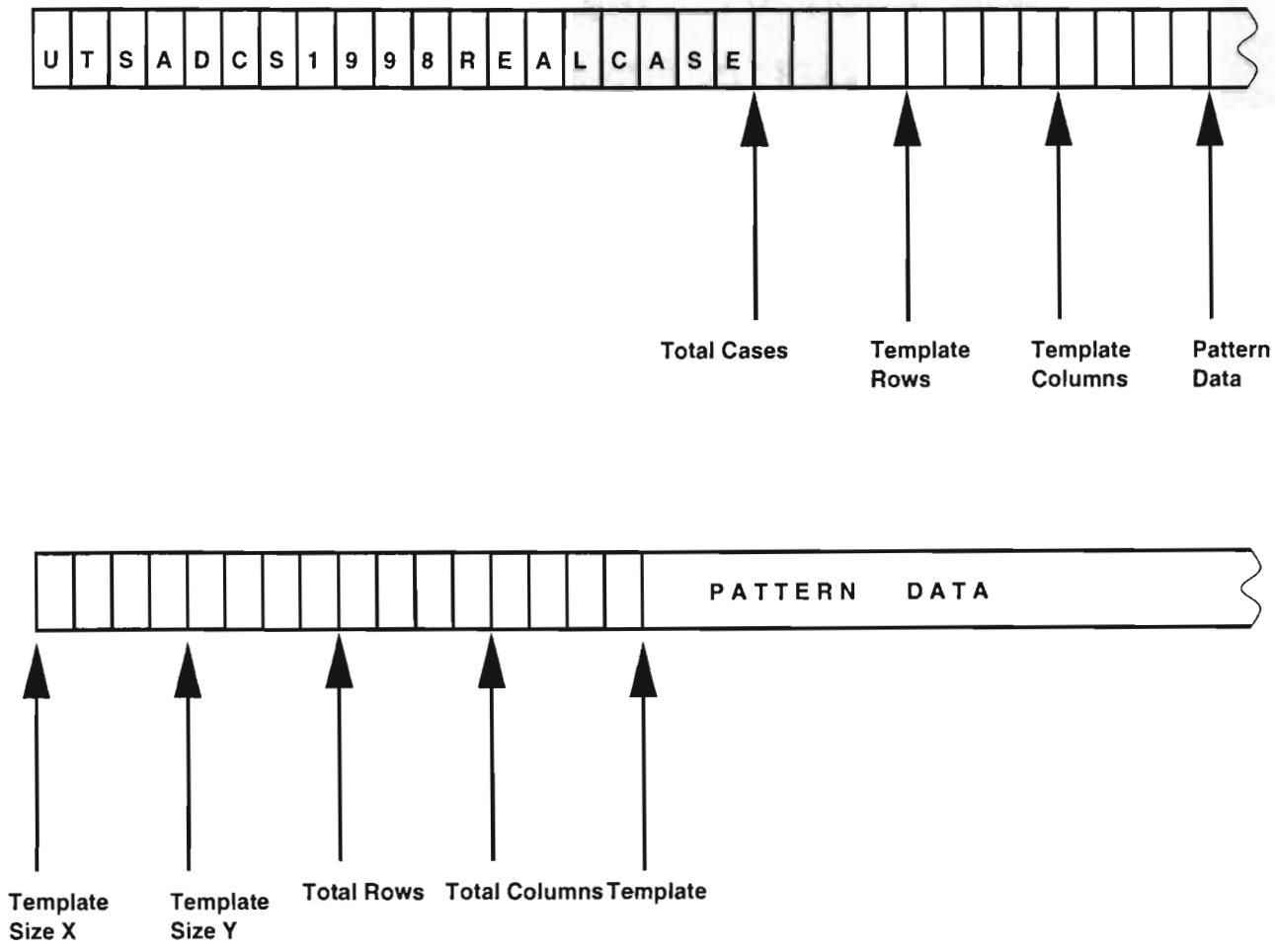

Figure 4.10: Real patterns database layout.

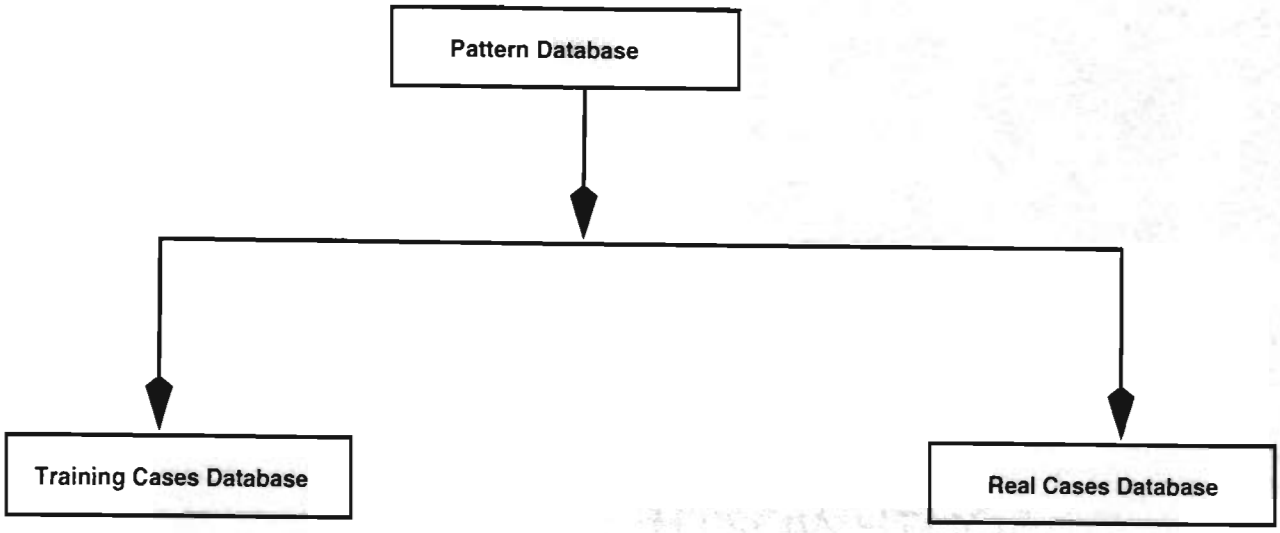

Figure 4.11: Object oriented model of pattern database management software component.

4.8.3 Pattern Database Management

Pattern database files require robust management software. Not only should it be fast and reliable, but also it should provide all functionality necessary to implement exponential pattern detection algorithms. A block schema of the pattern database management software is shown in Figure 4.12. It consists of the following modules: Memory Mapper, Physical Layout Verifier, Class Definitions, and a Pattern Manipulation module that comprises several submodules such as Pattern Query, Pattern Editing, and Pattern Insertion and Removal.

The database file is intended to grow very big over a period of time, since researchers continue to include new patterns to improve neural network response. The database is a critical element of this software package, since it affects performance and accuracy of the results. The mathematical operations over database patterns are highly repetitive and reuse patterns many times until all training conditions are satisfied. In order to provide the most efficient implementation, the Memory Mapper module is introduced in this package. This module uses operating system memory mapped file system calls as described in [26]. The entire database file is mapped

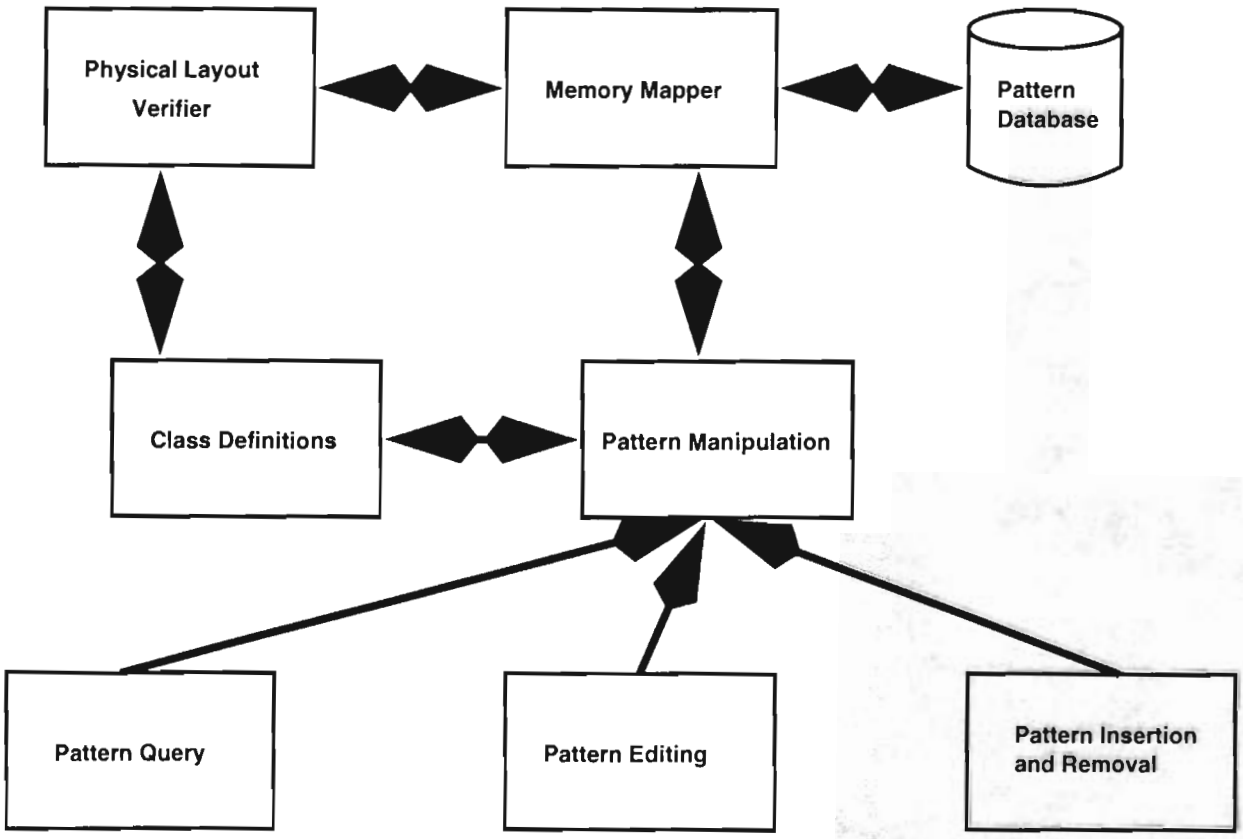

Figure 4.12: Block schema of the patterns database.

to RAM as soon as the database is open, however this memory space is not committed until it is requested. The operating system takes care of memory swappings. The performance gain is significant compared to sequential file access. When a database file is open, before it is mapped to memory, its layout is verified using the Physical Layout Verifier module.

Physical Layout Verifier provides a number of checks to make sure that a database file is not corrupt, and that the data have the correct format. Each separate check alone does not guarantee that the database is correct, however, taken together, they significantly reduce the probability that a corrupted database is used for core algorithms. The Physical Layout Verifier is implemented as an extendible module. Currently, it verifies the security descriptor, the corre-

spondence of the number of total cases in the database and the physical size of the database file, certain bytes and their values. For example, in the training database, a pattern data byte showing whether the pattern was marked for deletion may have a value of 0 or 1. If this byte has any other value, then it means that the database is corrupted. When a pattern is read, it is automatically mapped and committed to memory space using Class Descriptors. They are shown in Appendix **??**. Any operation over patterns is performed using Pattern Manipulation module. It consists of three submodules. When the pattern database is queried, the Pattern Query submodule is used. When a pattern should be edited, the Pattern Editor submodule provides necessary utilities to do that. Finally, when a pattern needs to inserted or removed from the database, the Pattern Insertion/Removal submodule operations are called. A simple object–oriented inheritance diagram is shown in Figure 4.11. The Pattern Database class is an abstract class that has two derived classes: Training Cases Database and Real Cases Database. Each class is responsible for handling the specific type of pattern databases.

4.8.4 Neural Network

The Neural Network component includes a core computational logic module and the auxiliary modules such as the Weights Loader. Before it is used, the neural network should be initialized. Weights Loader is used to read weights from a weight text file that has the following format: Set i, Weight $j : w_{ij}$ where i is a set number, j is a weight number, w_{ij} is a weight value. The Weight Loader parses this data to extract j, i, and w_{ij} and passes these values to the neural network class object. The object uses the values to initialize itself.

A block schema of a neural network is shown in Figure 4.13. It shows the central component that implements core computational logic and the data sources that it needs. The Core Component receives the weights from the Weight Loader and input data from a pattern database. The resulting value is produced by the core component. The implementation was done using object–oriented methodology. Initially, it was unclear which neural network to choose for the

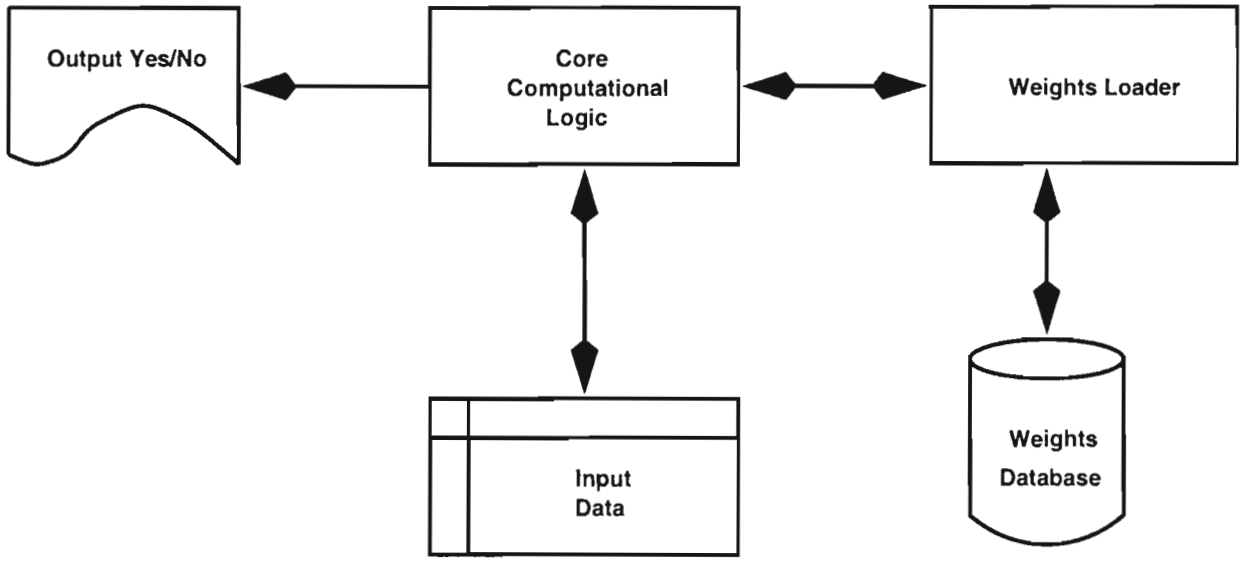

Figure 4.13: Block schema of neural network software component.

detection algorithm. The object–oriented analysis of this problem resulted in the solution that made the neural network component totally independent from the rest of the package. Through a simple interface a pointer to a memory array containing the data patterns is passed to the neural net object. This memory array is in fact a memory mapped database file. It allows the training procedure to run fast without adding a lot of overhead code.

4.8.5 Synthetic Pattern Generator

The purpose of using a neural network in this thesis was to recognize all exponentially looking patterns even in the presence of a low signal–to–noise ratio. These patterns are added to the training database from two sources. One source was mentioned already. It is a digitized differential energy data obtained from real dynamic systems. An analyst selects a subgraph pattern and inserts it into that database specifying whether the pattern is exponential. The second source is a synthetic pattern generator. Its object–oriented diagram is shown in Figure 4.14.

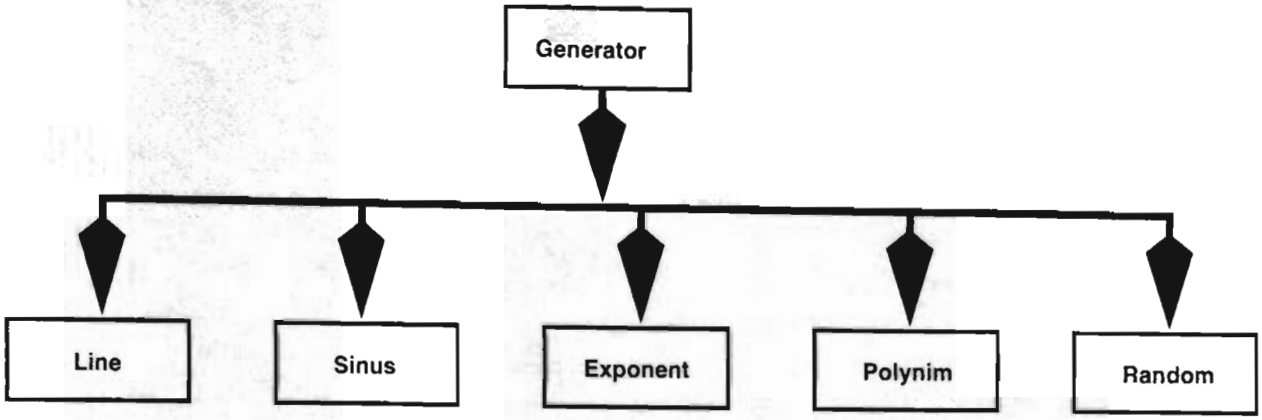

Figure 4.14: Syntetic pattern generator block schema.

The implementation C++ code is given in Appendix **??**. Base class CGenerator provides basic functionality that creates a set of data using mathematical formulae. These formulae are defined in the derived classes such as CSinusGenerator and CLineGenerator. When a synthetic data set is computed, it is passed to CDigitizer class object that digitizes it based on preset parameters.

4.8.6 NnetProcessor

The framework of NnetProcessor proposed in this thesis can be used for other projects that have to deal with training neural networks under varying requirements. Very few similar systems in existence today. One of them is called **NeuCOP Modeler** described in [27]. It is an integrated environment for processing data, modeling the dynamics of multi-variable processes, and simulating and testing the model. NeuCOP Modeler is designed for the process engineer responsible for the maintenance, operation, and optimization of multi-variable nonlinear processes. One of main modules in the exponential pattern detection software package is NnetProcessor. Its function is to maintain a training pattern database, provide a plug–in interface for neural network objects, classify data patterns, and train neural networks. It is a multithreaded, multifunctional, object–oriented application running under Windows NT and utilizing

Microsoft Foundation Classes library and various system features of this sophisticated operating system.

One of important features of NnetProcessor is the Graphic User Interface shown in Figure 4.15. An analyst can use a menu to intervene in an executing routine at any moment to provide a control action.

4.8.7 DataProcessor

Another main module in this software package is the DataProcessor used to load and show a differential energy graph. The DataProcessor allows an analyst to zoom in and out of different areas of the graph, select a digitizing window and view and edit digitized data. The most important functionality is semi–automatic and fully automatic detection of heteroclinic connections using the algorithms proposed in this thesis. For example, a researcher can select digitizing metrics, apply then to the differential energy graph, and start a semi–automatic exponential pattern detection procedure. When an exponential pattern is found it is shown to the researcher in a separate window. The researcher may proceed with the detection if he is satisfied with the found pattern, or insert it into the pattern database. If the latter is chosen, then the researcher may retrain the neural network on the fly with the newly added pattern, and continue the detection process. The Graphic User Interface for DataProcessor is shown in Figure 4.16.

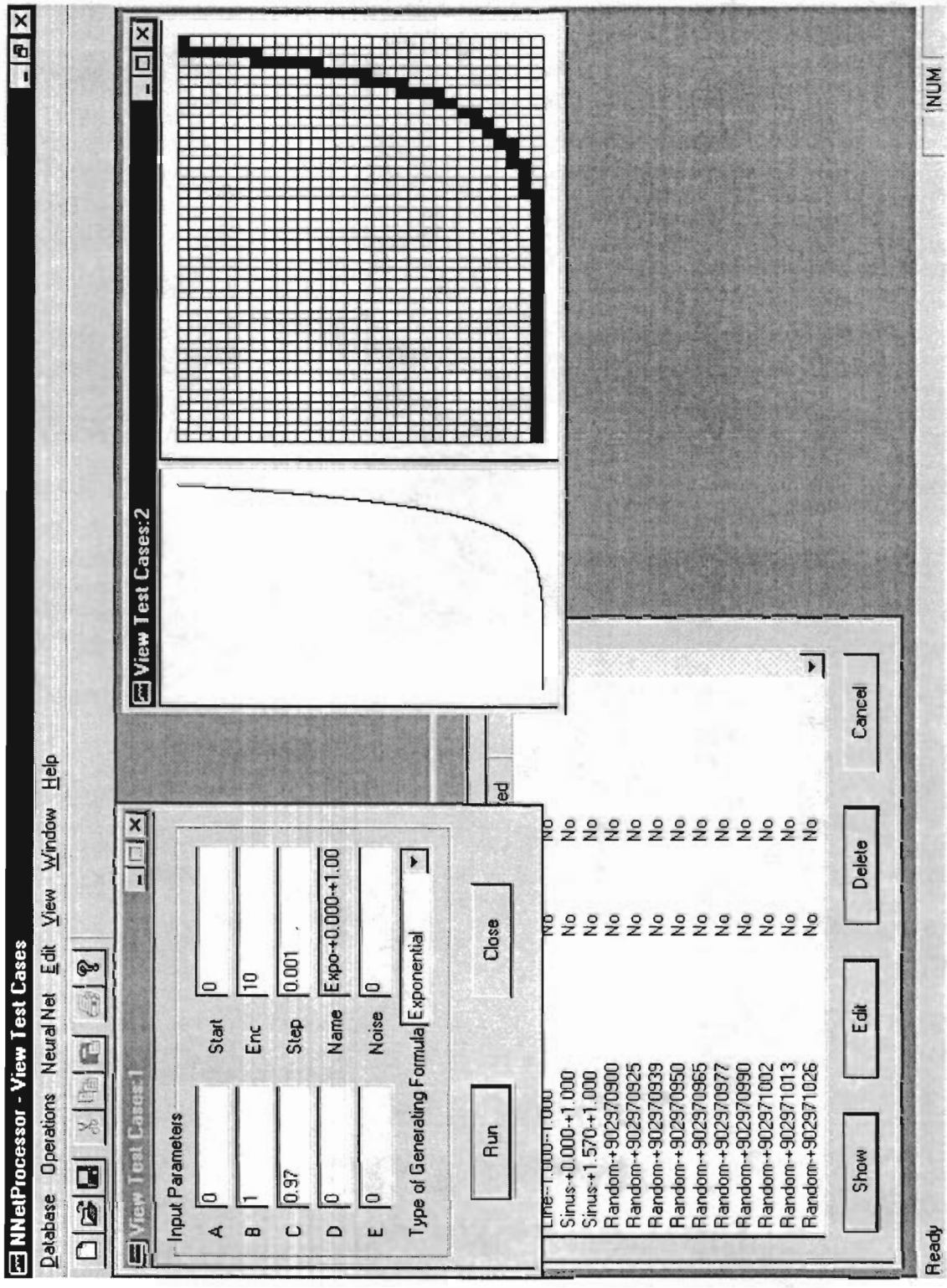

Figure 4.15: Graphic User Interface of NnetProcessor.

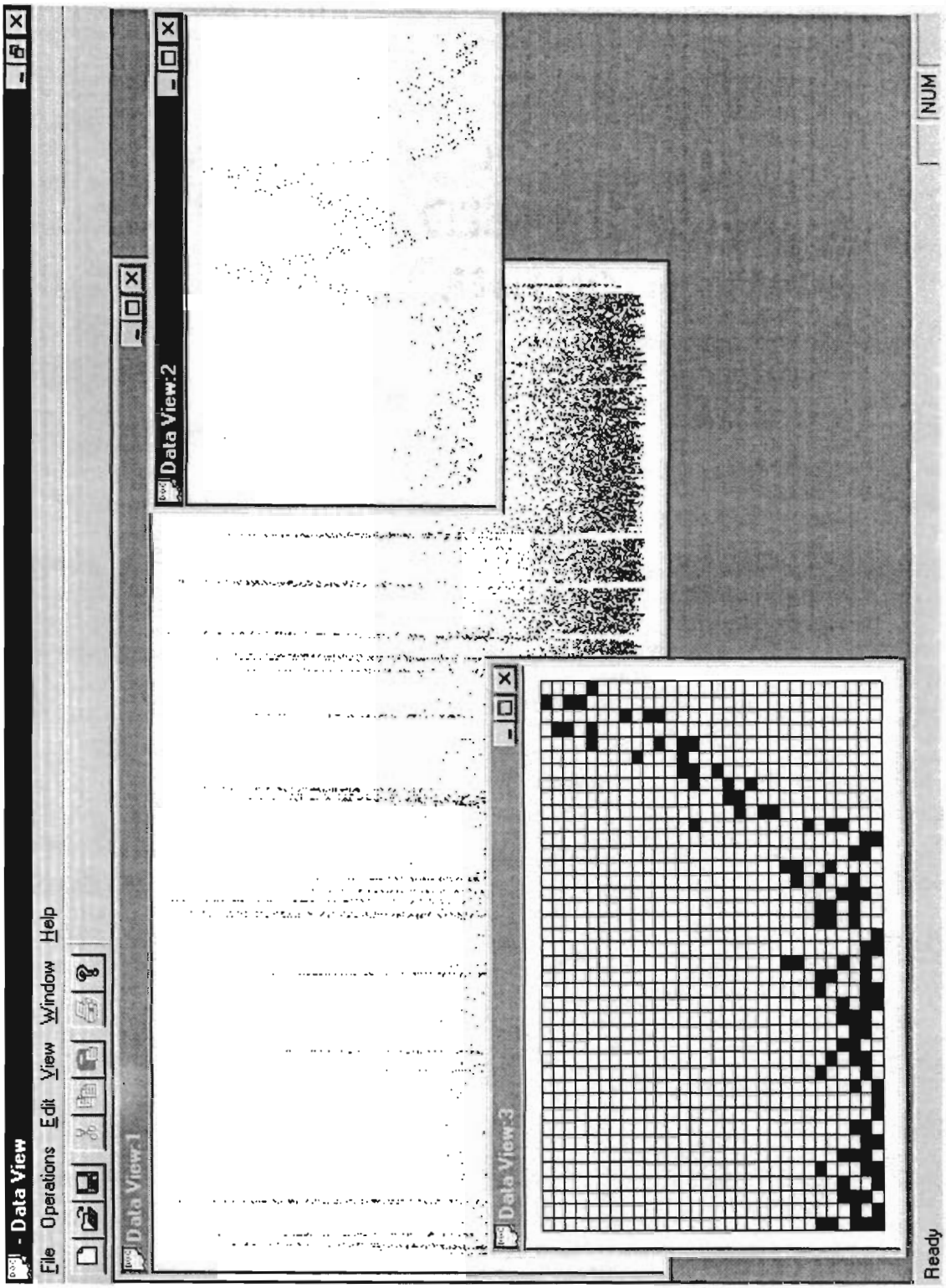

Figure 4.16: Graphic User Interface of DataProcessor.

Chapter 5

Experiments with Heteroclinic Differential Energy Data Sets

5.1 Introduction

In the previous chapters the Stone–Holmes theory was stated and verified through numerical experiments with Duffing and Ornstein–Uhlenbeck dynamic systems. However, the theory's usefulness can be confirmed only after experimenting with real physical systems. The first step in this direction was made in [10]. It was shown by performing a frame–by–frame analysis of a two–hour run that the histogram of passage times fits a single exponential curve. This work was the first demonstration that a skewed distribution of passage times with exponential tail detected in a real physical system corresponds to the expected signature of systems in which a trajectory is repeatedly injected near the stable manifold of a saddle in the presence of noise.

The results obtained by creating and running numerical simulation experiments in Chapter 3 showed very good correspondence with the Stone–Holmes theory. Real physical systems are much more complex to experiment with. Chapter 4 offers techniques and algorithms

that are used to collect data needed to apply to the Stone-Holmes theory. The first step is to obtain a differential energy time series of a dynamic system. This step is system dependent, and considered to be a preprocessing technique. Indeed, each system offers different physical properties, some of which may be selected for building differential energy time series. In the case of a real dynamic system an initial mistake in, for example, selecting an improper variable may lead to incorrect final results. It would therefore cast a shadow over the entire approach. This is the reason for using a mathematical model such as the Duffing equation to verify the validity of the techniques offered in Chapter 4. It constitutes the essense of the experiment described in Section 5.2 of this chapter. Sections 5.3 describes applications of the techniques on the combustion flame system.

5.2 Experiment with the Duffing Equation

An example of the graph of differential energy for the Duffing system is shown in Figure 5.1. A passage time interval is shown by vertical markers named t_a and t_b in Figure 5.1. Exponential growth and fall-off patterns are clearly seen. As the system is numerically simulated for an extended period of time and its trajectory stabilizes, the passage time becomes a constant number. It can be expressed by the formula

$$\lim_{n\to\infty}(t_i - t_j) = const \tag{5.1}$$

where t_i, t_j are time snapshots indicating the beginning of the exponentially declining pattern and the end of the following exponentially growing pattern, and n is discrete time.

Sample patterns of exponential growth and exponential fall–off are shown in Figure 5.2 and Figure 5.3 respectively. These patterns are characteristic for the graph of differential energy of the Duffing system.

When the digitization algorithm is applied to both graphs, their digitized equivalents are shown in in Figure 5.4 and Figure 5.5 respectively.

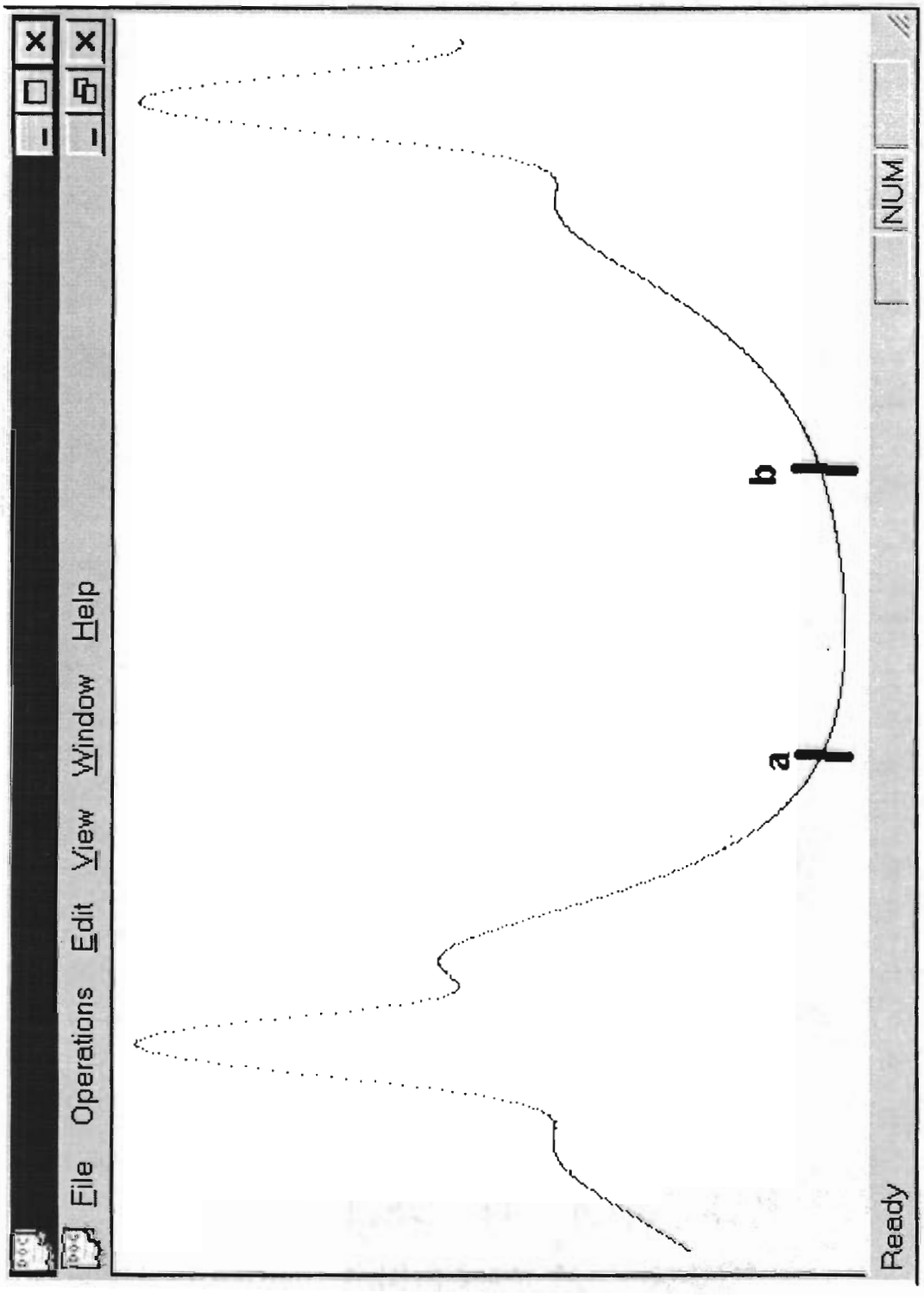

Figure 5.1: A sample graph of differential energy for the Duffing dynamic system.

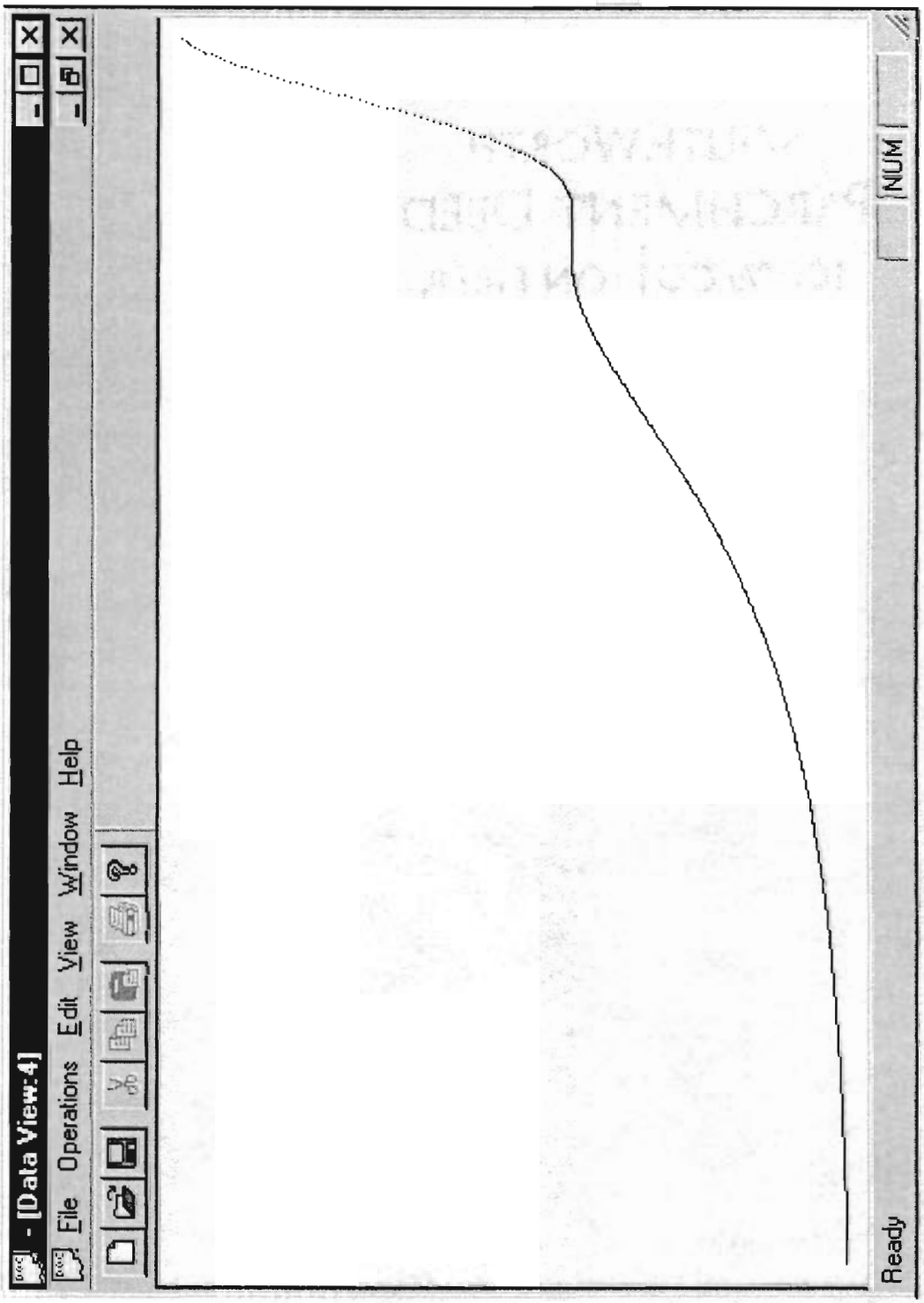

Figure 5.2: A sample pattern of exponential growth in the graph of differential energy for the Duffing dynamic system.

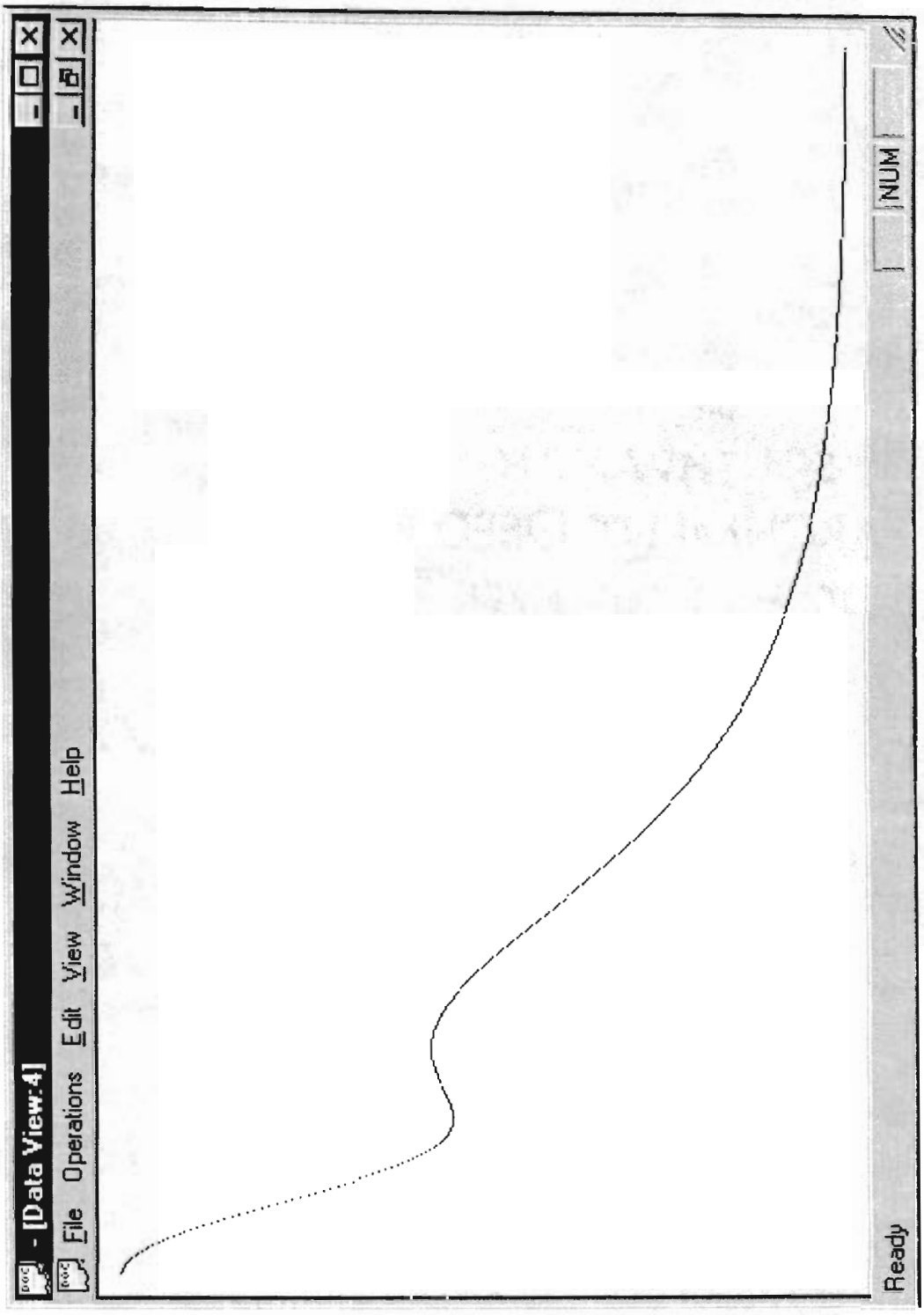

Figure 5.3: A sample pattern of exponential fall–off in the graph of differential energy for the Duffing dynamic system.

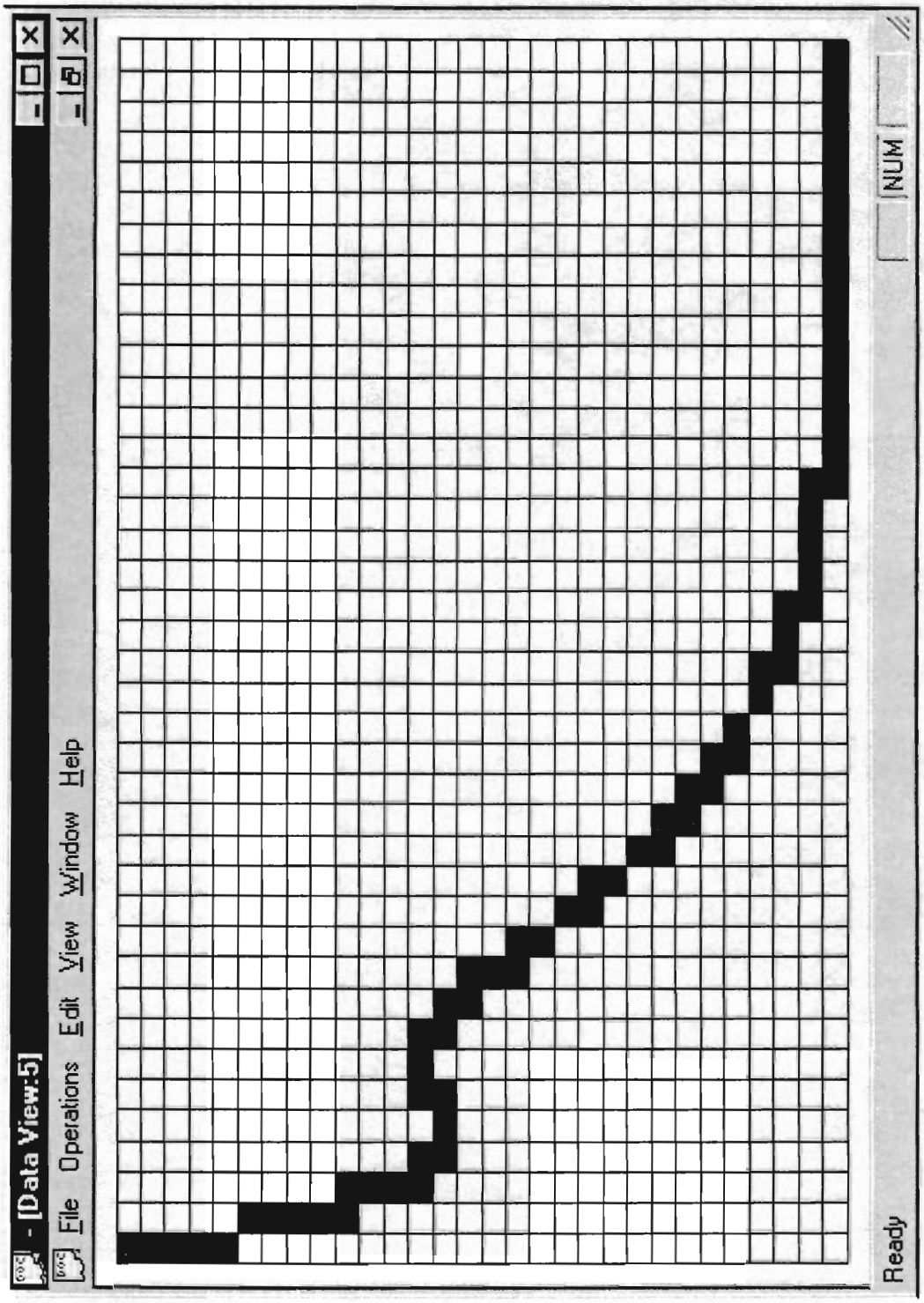

Figure 5.4: A sample pattern of digitized exponential fall–off pattern in the graph of differential energy for the Duffing dynamic system.

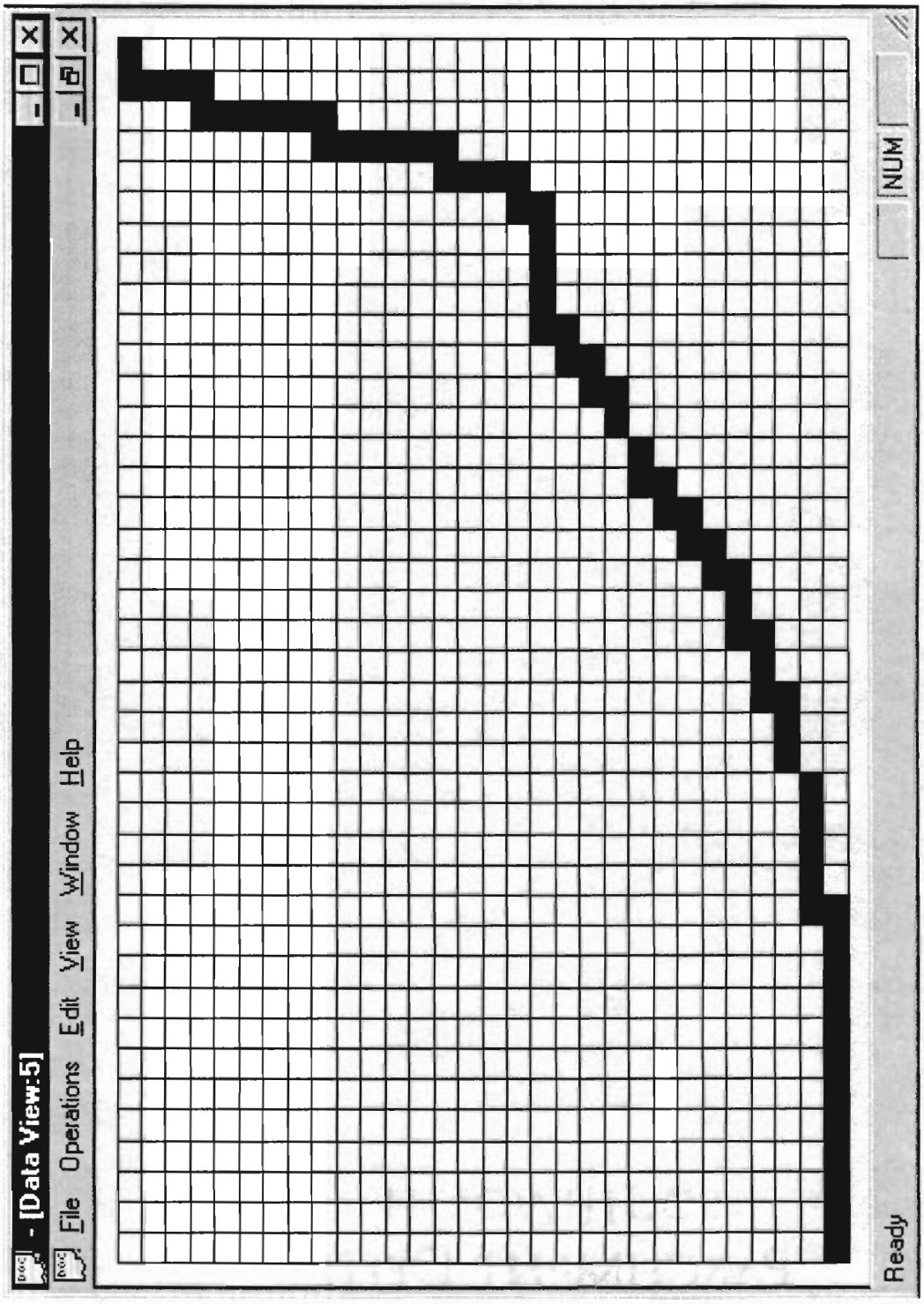

Figure 5.5: A sample pattern of digitized exponentially growing pattern in the graph of differential energy for the Duffing dynamic system.

When noise is applied to Equation 3.5 then the patterns in the differential energy graph for the Duffing equation will vary in shape, amplitude, and passage time. At this point, it makes sense to run the experiment over the Duffing system with noise to build a passage time histogram. The system was simulated with noise amplitude $\epsilon = 0.03$. The passage time histogram is shown in Figure 5.6.

Table 5.1 shows and compares the resulting probabilities for experimental data obtained by the box experiment, and using the the differential energy data set. The probability distribution data obtained by the box experiment is shown in Table 3.1. Parameters for the Duffing model **are given** in Section 3.7.

The first column of Table 5.1 "Passage Time Interval" shows the right border of each subinterval. The second column, "Probability", is divided into two subcolumns. The first subcolumn displays the probability obtained in the box experiment. The second subcolumn displays the probability obtained by the experiment with the differential energy data set. The last column, "Deviation" shows the percentage of the difference between the two probabilities for each row relative to the first probability value. Analysis of this column determines that 50% of all deviations between the two experimental probabilities data sets are less than 5%, and 90% are less than 10%. The average deviation is equal to 6.6%. It proves the validity of the proposed algorithm at least on the Duffing system.

5.3 Experiment With the Flame Data Set "Original"

A graph of differential energy for a combustion flame system is shown in Figure 5.7. A description of this experiment is given in [18]. A movie of premixed flames was taken for two hours. The propane–air flammable mixture was supplied to the porous plug burner at a flow rate of 8.27 liters/min with equivalence ratio of 1.85, and a pressure of $\frac{1}{2}$ atm. Researchers analyzed the movie to detect all exponential patterns and build the probability histogram. The analysis took several months. This is important to know this in order to compare it with the time that is

Passage Time Interval	*Probability*		*Deviation, %*
1	0.0	0.0	0
2	0.003	0.0	n/a
3	0.0	0.00973	n/a
4	0.0	0.0	n/a
5	0.0	0.0	0
6	0.0113	0.0107	5.31
7	0.003	0.0	n/a
8	0.01752	0.01653	5.65
9	0.0598	0.062	3.67
10	0.1652	0.149	9.8
11	0.2201	0.2086	5.22
12	0.1811	0.1748	3.4
13	0.1428	0.1326	7.14
14	0.1024	0.1097	7.13
15	0.0169	0.0059	65.1
16	0.0502	0.0311	38.1
17	0.0237	0.0221	6.8
18	0.003	0.	n/a
19	0.0	0.003	n/a
20	0.0	0.003	n/a

Table 5.1: Probability distribution values for Duffing ODE box experiments and Duffing Differential Energy experiment.

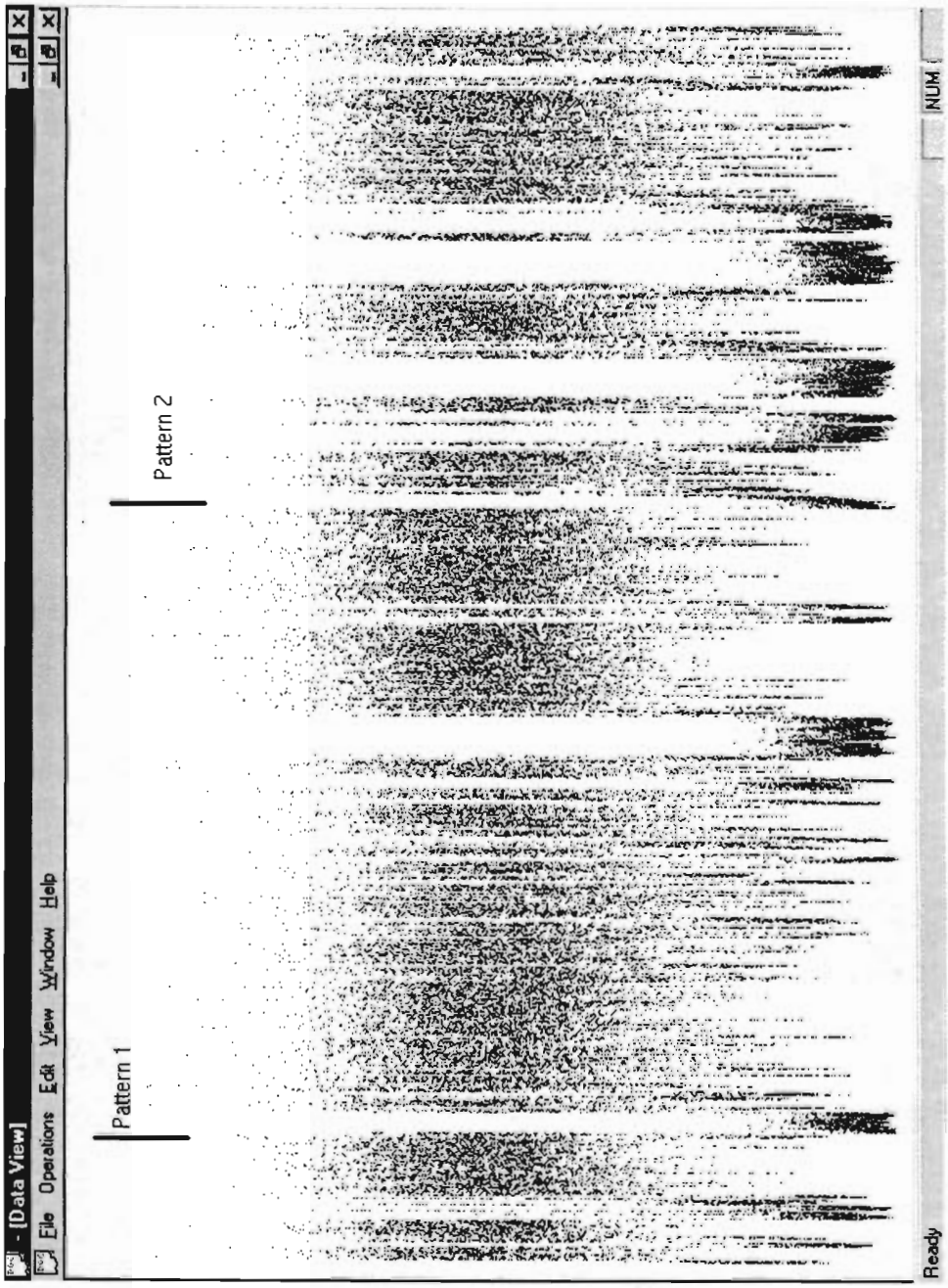

Figure 5.7: A sample graph of differential energy for "Original" data set.

taken by **DataProcessor** to analyze the same run. The difference is stunning - it took only 27 minutes to find all the exponential patterns. The two–hour run provided data to build a differential energy graph for 60, 000 entries. In the graph in Figure 5.7 two patterns were selected randomly that correspond to manually detected exponential patterns. The zoomed windows containing the patterns are shown in Figure 5.8 and Figure 5.9. In both cases one can see that exponential patterns are represented by a well on both graphs relative to the average value of differential energy. Every connection is preceeded by an exponentially declining curve that stabilizes on the bottom of the pattern. A connection is ended with an exponentially growing curve that is the other end of the well. One can see that the differential energy graph shown in Figure 5.7 has a strong stochastic component despite the presence of a strong exponential constituent. It makes the task of recognition of an exponential pattern even more challenging. The digitizing technique helps to reduce the level of noise by "consuming" the stochastic component. The digitized graph for an exponential pattern from "original" data set is shown in Figure 5.10. It does not look as pretty as a digitized patterns from the differential energy graph of the Duffing system shown in Figure 5.4 and Figure 5.5. However, the noise–tolerant nature of neural networks provides the necessary sensitivity to detect exponential components of patterns. In order to verify whether detected exponential patterns are truly heteroclinic, additional information is provided to the **DataProcessor**. This information has to do with the specific spatial organization of flame patterns. This organization shows concentric rings of cellular flames which appear and disappear at irregular intervals. By counting the number of cells on the inner and outer ring and the number of frames during which the pattern is stable, a valuable data is provided to verify the validity of the detected exponential patterns. After running **DataProcessor** over the data set with the additional verification algorithm, only 17% of all the detected exponential patterns were ruled out as nonheteroclinic. The resulting probability histogram for the "original" data set is shown in Figure 5.11.

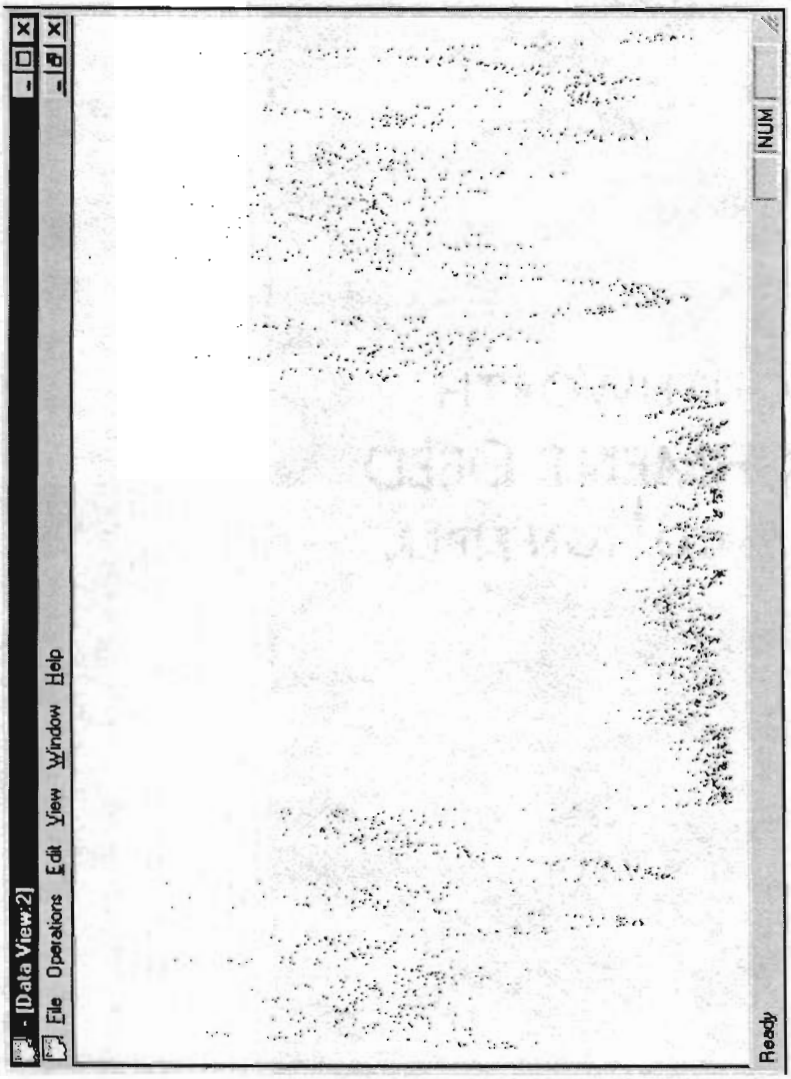

Figure 5.8: A selected window for pattern 1 from the "Original" data set.

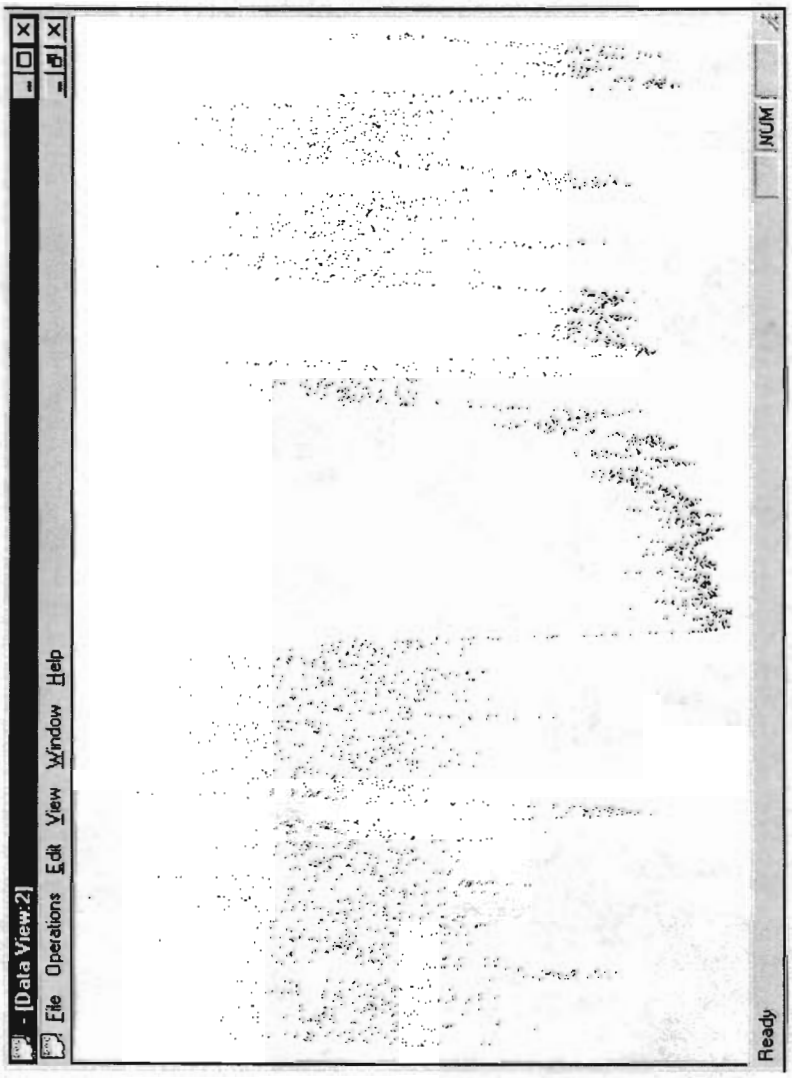

Figure 5.9: A selected window for pattern 2 from the "Original" data set.

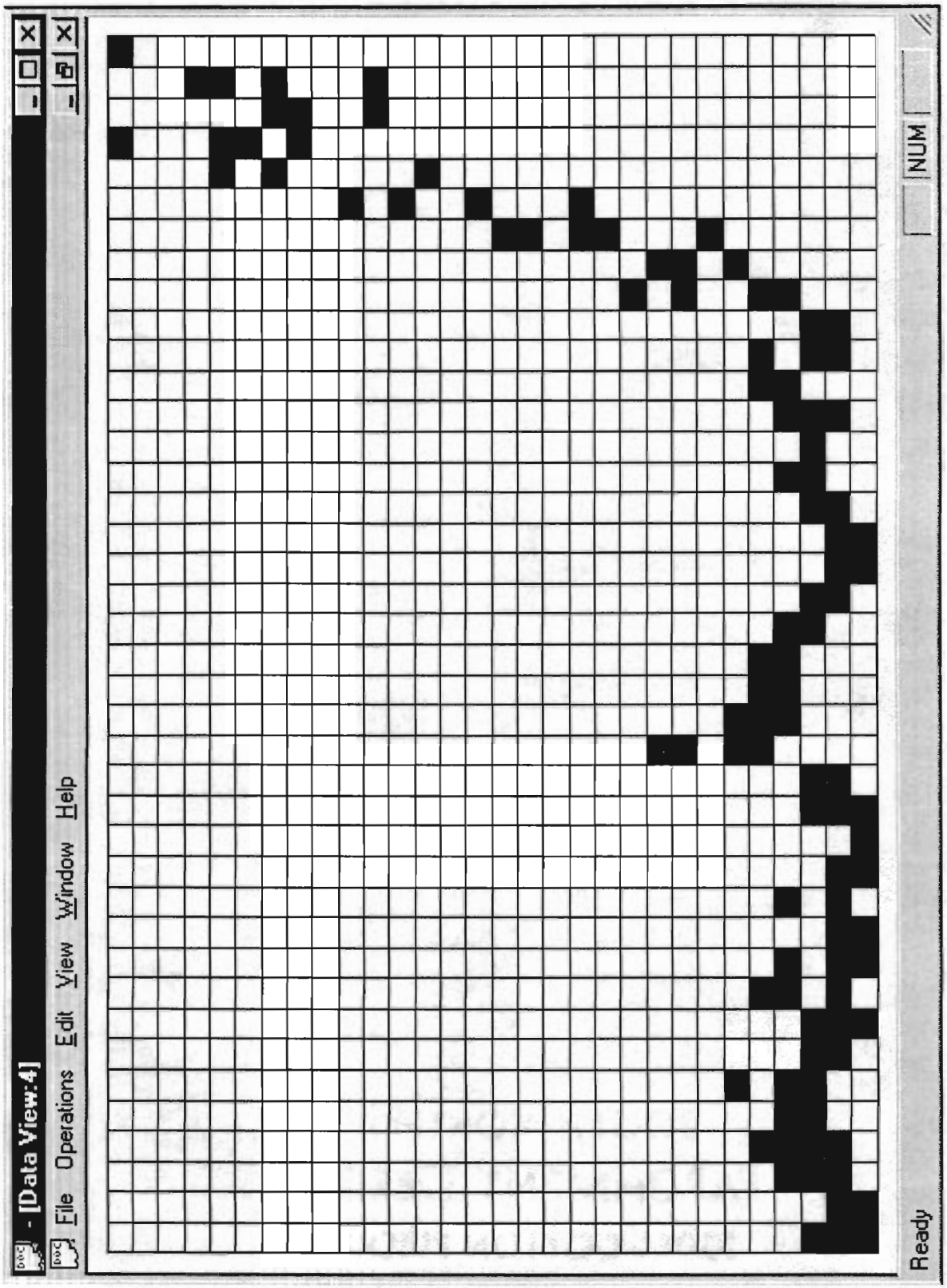

Figure 5.10: Digitized pattern 1 from the "Original" data set.

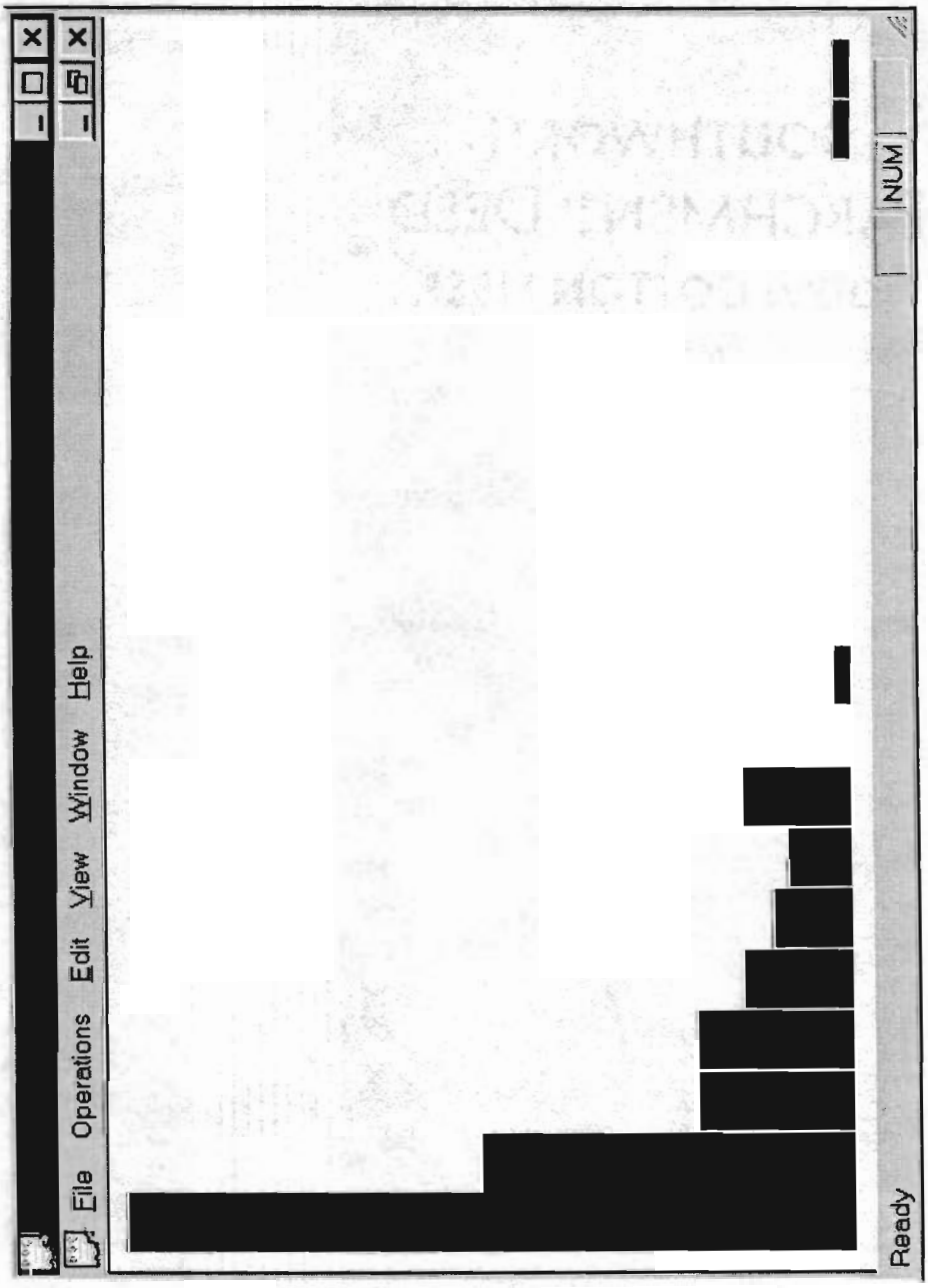

Figure 5.11: The probability histogram build from the "Original" data set.

5.4 Experiments With Other Data Sets

In this section four figures are included. Several movies were taken for flame experiments carried on under two different sets of parameters. The differential energy graphs are different in both cases although they all have similar patterns that are used by the detection techniques developed in this thesis. As a result of applying these techniques the probability histograms are built. Both histograms have very strong exponential component.

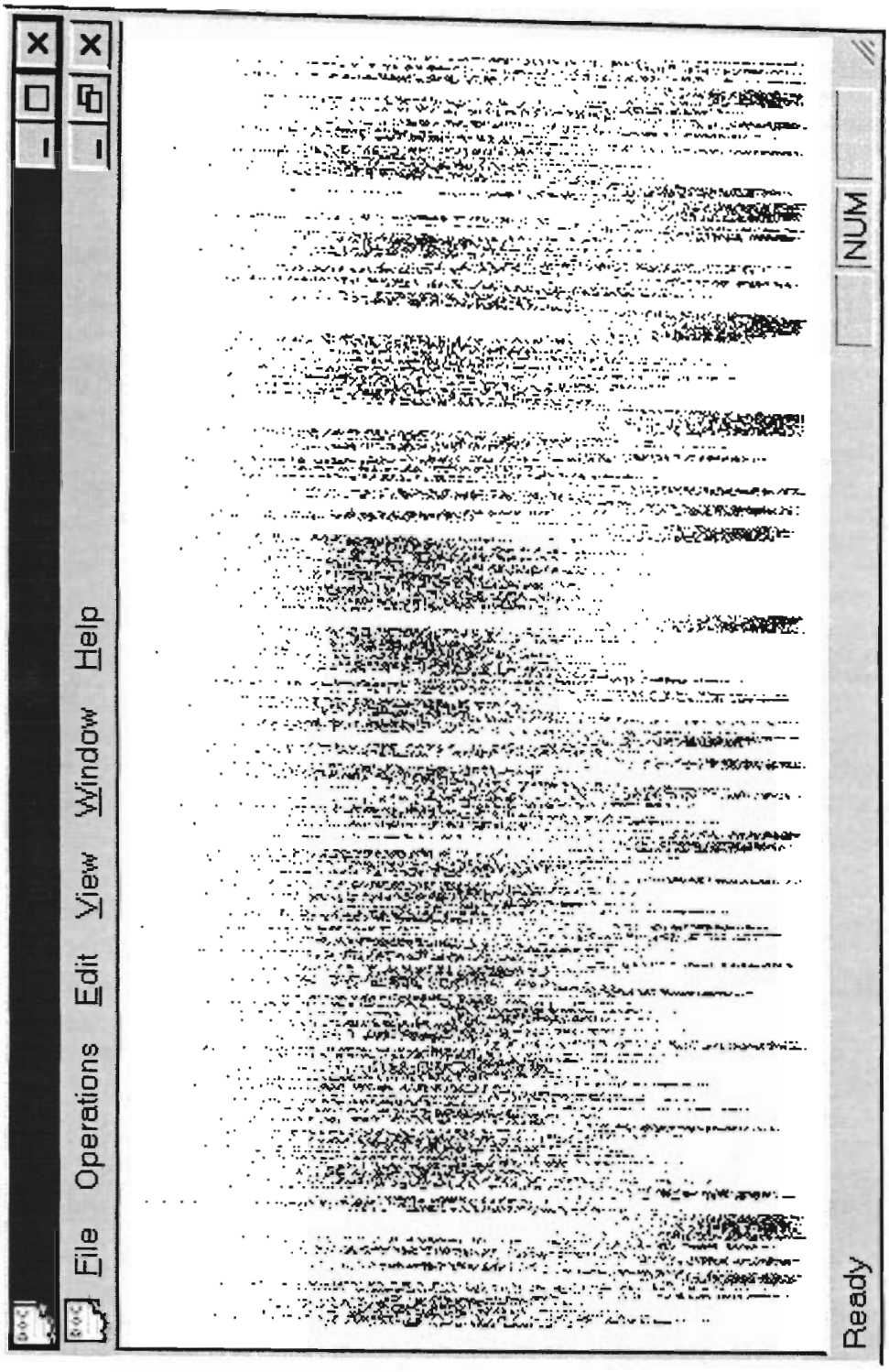

Figure 5.12: Differential energy graph build for a flame run "HeteroA".

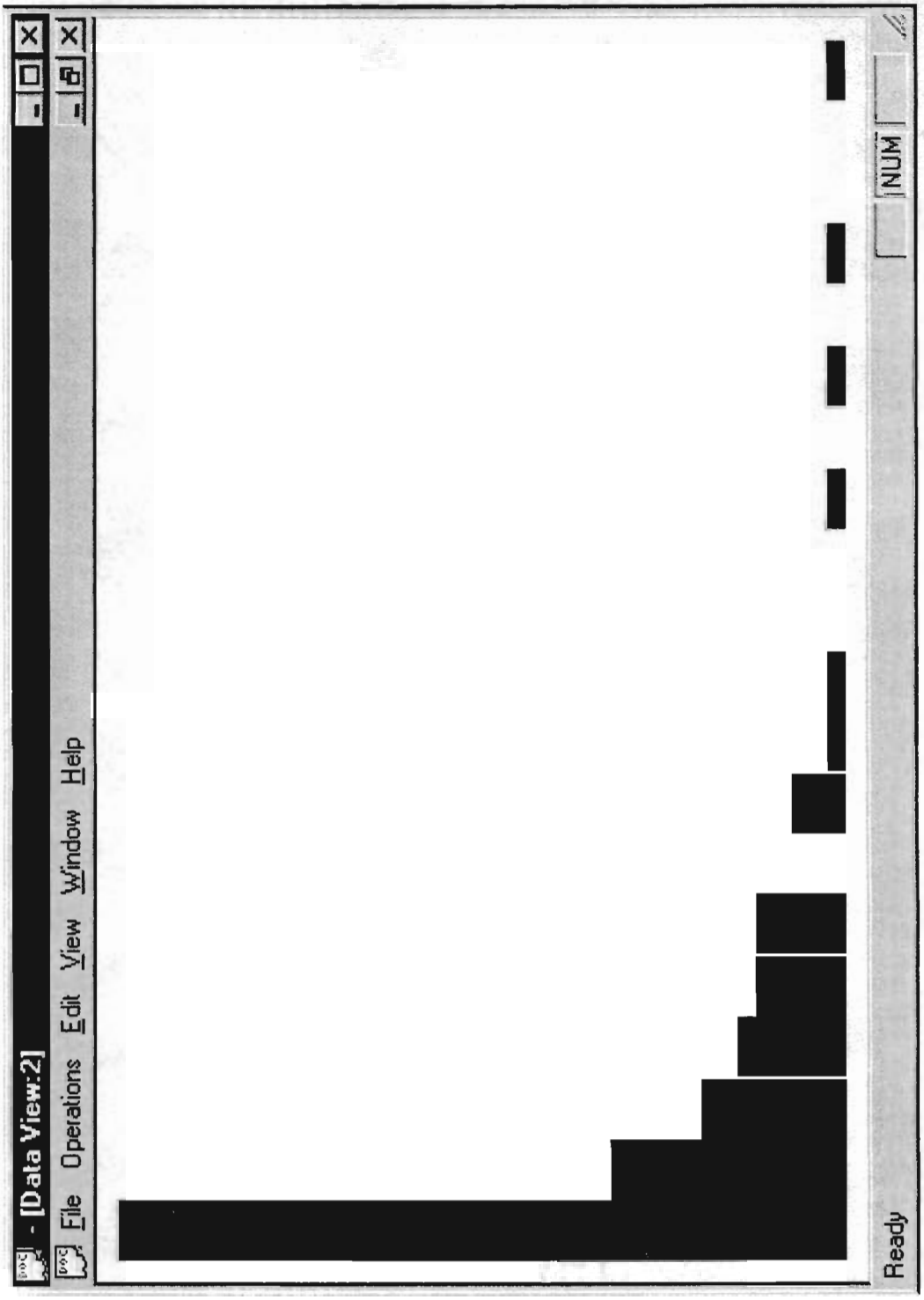

Figure 5.13: The probability histogram build from "HeteroA" data set.

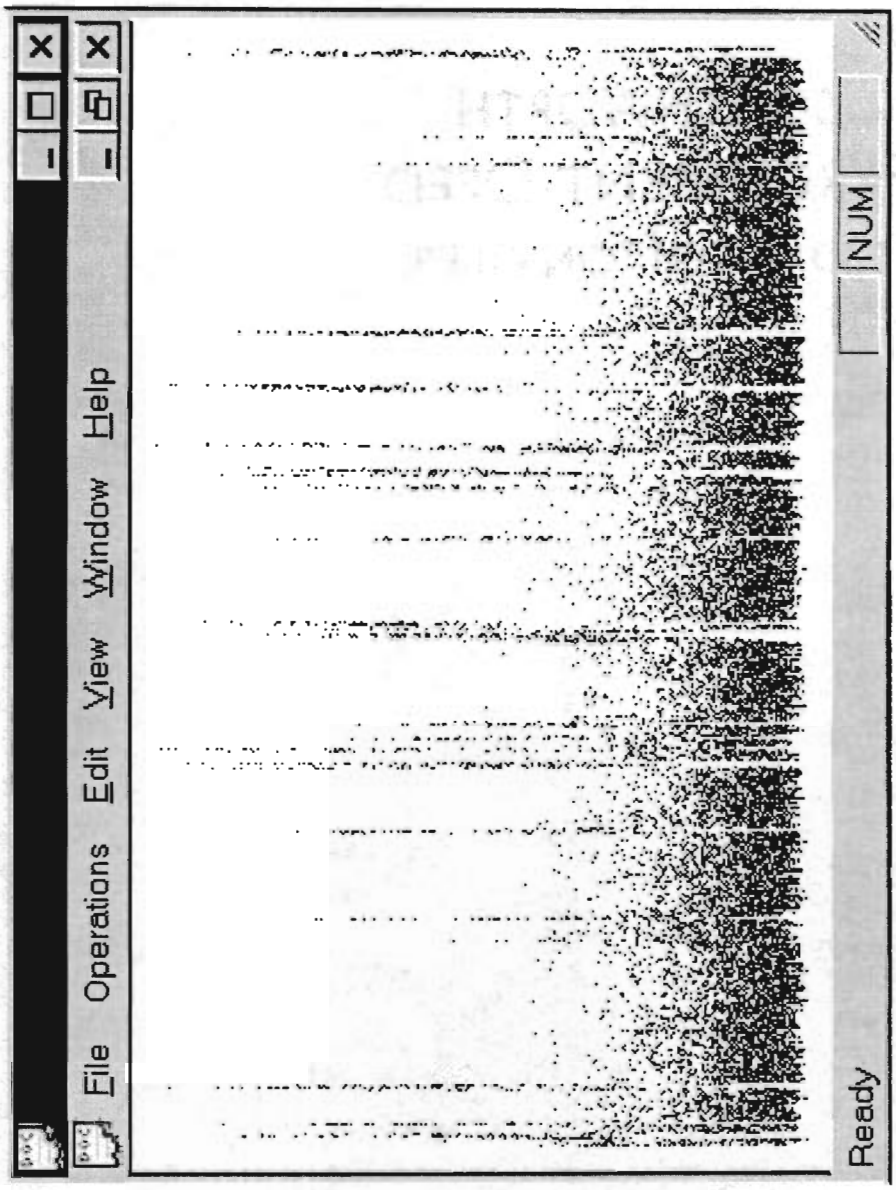

Figure 5.14: Differential energy graph build for a flame run "HeteroC".

Chapter 6

Conclusions

The major achievement of this thesis is the creation of a technology that can be applicable to a wide range of problems. The generalization of the task of detecting exponential patterns in experimental data yields the more common task of detecting a predefined pattern in time series data. Some patterns have very complex structure, and occur in a small range of a data set. The straightforward approach of using an analyst to review such data set with the purpose of detecting all patterns takes a lot of time. The analyst does the "compression" by discarding the majority of data that does not fit the characteristics of the desired pattern. Using various computational algorithms such as curve fitting and Fourier analysis is costly as well. These algorithms are designed and fine–tuned for a specific pattern. They use computationally expensive operations and perform poorly in the presence of low signal–to–noise ratio. If characteristics of the model pattern are changed then the algorithms need to be adjusted appropriately. The approach proposed in this thesis is free from all mentioned drawbacks. Not only is it fast and reliable, but also it has the inherent quality that its accuracy can be improved on the fly in a real–time environment. Researchers create a database in which they store model patterns. This database is used to train a neural network that is a central part of the software package developed in this thesis. The pattern database may be updated at any time with new patterns in order to retrain

the neural network. It means that every time the software is used for a data set, the quality of recognition of the neural network is improved. One of the basic requirements that prompted the use of the techniques proposed in this thesis was the necessity to recognize patterns that only remotely resemble some characteristics of the model patterns. It is still easier for the analyst to discriminate several recognized patterns than to scan through the entire data set.

A number of computer techniques and algorithms for detecting heteroclinic connections in flame images were designed, implemented, and applied to the Duffing dynamic system and a real combustion flame system. The resulting probability histograms were used to verify whether the Stone–Holmes theory is applicable to different dynamic systems. This task was solved in several major steps

1. Analyze Stone–Holmes theory.
2. Apply the theory to a theoretical model with heteroclinic connections.
3. Build the probability histogram and show its conformity with the theory.
4. Analyze the physical system and find characteristic parameters.
5. Compute the differential energy data set from the characteristic parameters.
6. Apply the digitization technique to tranform analogous differential energy data into a digital array.
7. Use a neural network trained to recognize exponentially growing and declining patterns at the end and beginning of a suggested exponential pattern.
8. Use additional data for verification of the heteroclinic connections.
9. Build the resulting probabilities histogram.

Research in the field of dynamic systems is extremely difficult when it comes to analyzing heteroclinic and homoclinic connections. Generally speaking, these connections are structurally unstable. If system's parameters are changed infinitesimally, then the connection

ceases to exist. In real physical dynamic systems, such connections are impossible to detect since it is difficult to maintain constant system parameters. However, the presence of symmetry groups under which a vector field is equivariant can induce structurally stable orbits. Stone–Holmes theory showed that the time distribution during which a trajectory of such a dynamic system stays within a point of stable equilibrium is governed by Equation 2.9.

The Stone–Holmes theoretical result is the starting point for the work done in this thesis. After each component of the theory such as the Wiener noise process, the Ornstein–Uhlenbeck process, etc. were analyzed, numerical experiments with the Ornstein–Uhlenbeck system and the Duffing equations were designed and carried out. These experiments showed how the models' behavior was influenced by external conditions such as noise. In order to conduct these experiment a software package was developed in X–windows. This package comprised a number of algorithms, standard and modified, that produced the results needed to show that Stone–Holmes theory is correct for these numerical systems.

The next step was to analyze a real physical system, and apply the theory to verify its validity for a dynamic system. Such a system was created in University of Houston. It was a spatio–temporal premixed flame on a porous plug burner which was housed in a combustion chamber made from process glass pipe. This system produced a rich set of spatially organized flames. Although most flame patterns were classified in [18], there were no techniques that allowed a researcher to detect automatically the beginning and the end of every heteroclinic connection.

Generally speaking, there are hardly any standard techniques that can be used to analyze spatio–temporal behavior of dynamic systems. The techniques developed in this thesis are only partially unique since they were applied to a specific system. They are general since there are many systems that can be tranformed to those having similar basic behavior and functionality. The software package that was developed in this thesis can be used for dynamic systems where the borders of specific behaviors are outlined by clearly visible patterns. There are several directions for future research in this area. The physical domain contains an infinite number of

various spatio–temporal patterns that needs to be analyzed and selected from a huge source data array. Improvements of the proposed algorithms and techniques can be offered for the neural network component as well as for verification procedures. For example, an adaptive algorithm can select, or even switch between different neural networks depending on the complexity and type of recognized patterns. Various verification algorithms can be added to the existing implementation with the purpose of checking the validity of a selected pattern. Additional components can be added that extract numerical characteristics for recognized patterns, for example calculate eigenvalues for recognized exponential patterns.

Bibliography

[1] E. Stone and P. Holmes, Random perturbations of heteroclinic attractors, *SIAM J. Appl. Math.*, **50**, No.3, 1990, p. 726–743.

[2] V. Melnikov On the instability of the center for time periodic perturbations, *Trans. Moscow Math. Soc.*, **12**, (1), 1969, p. 1–57.

[3] N. Aubry, P. Holmes, J. Lumley, and E. Stone, The dynamics of coherent structures in the wall region of a turbulent boundary layer, *J. Fluid Mech.*, 1988, **192**, p. 115–173.

[4] E. Stone, Unstable fixed points, heteroclinic cycles and exponential tails in turbulence production, *Physics Letters A*, **155**, (1), 1991, p. 29–42.

[5] F. Busse and K. Heikes, Convection in a rotating layer: a simple case of turbulence, *Science*, **208**, 1989, p. 173–175.

[6] F. Busse, Transition to turbulence in Rayleigh–Benard convection, *Hydrodynamic instabilities and the transition to turbulence*, ed. H.Swinney and J. Gollub, Springer-Verlag, 1981.

[7] J. Guckenheimer and P. Holmes, *Nonlinear oscillations, dynamical systems and bifuractions of vector fields*, Springer-Verlag, Berlin, New York, 1983.

[8] R. May, Simple mathematical model with very complicated dynamics, *Nature*, **261**, 1979, p. 459–467.

[9] Y. Pomeau and P. Manneville, Intermittent transition to turbulence in dissipative dynamical systems, *Commun. Math. Phys.*, **74** 1980, p. 189–197.

[10] E. Stone, M. Gorman, M. el-Hamdi, and K. Robbins, Identification of Intermittent Ordered Patterns as Heteroclinic Connections, *Physical Review Letters*, **76**, (12), 1996, p. 48–57.

[11] E. Jackson, *Perspectives of nonlinear dynamics*, Cambridge, Univ. Press, 1995.

[12] P. Todorovic, *An introduction to stochastic processes and their applications*, Springer-Verlag, New York, 1993.

[13] L. Breiman, *Probability*, Addison–Wesley, Reading, MA, 1968.

[14] R. Gonzalez and R. Woods, *Digital Image Processing* Addison-Wesley, Reading, Mass., 1993.

[15] Z. Nitecki, *Differential Dynamics*, M.I.T. Press, 1983.

[16] F. Moon, Experimental models for strange attractor vibrations in elastic systems, *in New approach to nonlinear problems in dynamics*, ed. P. Holmes, pp. 487-495, SIAM.

[17] J. Mohqmed, *Numerical Algorithms*, Oxford, Clarendon Press, 1986.

[18] M. el-Hamdi, M. Gorman, and K. Robbins, *A picture book of dynamical modes of flat, laminar premixed flames*, 2nd edition, Houston and San Antonio, October 1990.

[19] A. Weigend and N. Gershenfeld, *Time series prediction*, Proceedings, **XV**, Addison-Wesley, Reading, MA, 1993.

[20] E. Stollnitz, *Wavelets for computer graphics*, Morgan Kauffmann Publishers, Inc., San Francisco, 1993.

[21] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation, Lecture Notes Volume I*, Addison-Wesley, Reading, MA, 1996.

[22] L. Fausett, *Fundamentals of Neural Networks*, Prentice Hall, New Jersey, 1994.

[23] B. Widrow and M. Lehr, 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation, *Proceedings of the IEEE*, **78**, (9), p. 1415–1442.

[24] R. Hecht–Nielsen, Theory of the backpropagation neural network, *International Joint Conference on Neural Networks*, Washington, DC, **1**, p. 593–605.

[25] J. Morril, Distributed recognition of patterns in time series data, *Communications of the ACM*, **41**, (5), p. 45–51.

[26] M. Pietreick, Internals of Memory–Mapped Services in Windows NT 4.0, *Microsoft System Journal*, (12), 1997, p. 23–49.

[27] Internet marketing guide. *http://neuralware.com/products/neucop/ncover.html*

[28] K.A. Robbins, A new approach to subcritical instability and turbulent transitions in a simple dynamo, *Math. Proc. Camb. Phil. Soc.*, **82**, 1977, p. 309–325.

Vita

Mark Grechanik was born on December 4, 1966 in Zhitomir, Ukraine, USSR. He received a M.S. degree in Electrical Engineering from the Kiev Polytechnic University in 1991. He received his first patent for "Cable Inlet and Outlet" and started his professional carrier at age 18. He worked as Senior/Lead/Principal Software Engineer/Consultant for a number of high–tech companies in the USA and abroad, such as IBM Corp, Arrowsmith Technologies Inc., UniSQL Inc., and others. His current recent interest include distributed computing and expert systems. He married Tina Grechanik in 1991.

Permanent address: 11215 Research Blvd Apt. 2165
Austin, TX 78759

This thesis was typeset by the author in LaTeX.

COMPUTER TECHNIQUES AND ALGORITHMS FOR DETECTION OF HETEROCLINIC CONNECTIONS IN EXPERIMENTAL DATA

by

Mark Grechanik, M.S.

THESIS

Presented to the Graduate Faculty of
The University of Texas at San Antonio
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Science and Engineering
Division of Computer Science
November 1998

COMPUTER TECHNIQUES AND ALGORITHMS FOR DETECTION OF HETEROCLINIC CONNECTIONS IN EXPERIMENTAL DATA

APPROVED BY SUPERVISING COMMITTEE:

Dr. Ray A. Robbins, Chair

Dr. Thomas C. Bylander

Dr. Steven Robbins

Dr. Mary Lou Zeeman